Cena 9900 zł

Indeks 379972

tajemnice ATARI

4 (12)/92

PISMO UŻYTKOWNIKÓW ATARI XL/XE

W NUMERZE:

Mapa pamięci 11-12

Big Clock 3
Zegary 6-7

AB...

Piszemy Demo 13-14
Forth (cz. 4) 19-22

BASIC DOS 4
Mini-BATCH 8
TRACE 15
TOS 23-26

Łata do ACTION! 9

Prawdziwi Programiści 28-31

Voo Doo Castle 16-18
Klocki 26-27

Nowe gry 8

Ludzie listy piszą 9

ZA MIESIĄC:

- UJAWNIAMY SEKRETY ATARI BASIC
- W CYKLU ATARI INACZEJ ODBIÓR OBRAZÓW PRZEZ RADIO
- KULKI (GRA DO WPISANIA)

Drodzy Czytelnicy!

Dziękuję Wam za owocną współpracę, proszę o jeszcze! Już dziś ponad połowa objętości TA pochodzi z nadesłanych materiałów. Nic nie stoi na przeszkodzie, by cała gazeta była kreowana przez jej czytelników. Dotyczy to nie tylko konkursów. Jeżeli napisałeś program, który zajmuje 6 linii (albo 10, 100, czy nawet 1000), nie frasuj się. Przyślij go do nas, obejrzymy, a jeśli jest tego wart – wydrukujemy. Każdy autor, którego dzieło jest publikowane w TA, otrzymuje należne honorarium. Jeżeli program nie pasuje do ram konkursów (5 linii, 6. strona), to kwota może być co najwyżej mniej okrągła.

Nie przysyłajcie programów przepisanych z innych czasopism! Jest to bardzo poważny występek i wobec stosujących tego typu praktyki będziemy występować z całą surowością.

Wiele osób pyta: jak pisać do TA? Oczywiście, przede wszystkim, z sensem. Nie stawiamy żadnych tematycznych ograniczeń. Najwyżej oceniane są takie opracowania, które odkrywają przed czytelnikiem coś nowego. Wątpię, czy takim odkryciem może być fakt, że jeśli zmieniać sprytnie kolory na ekranie, to powstaje efekt tęczy.

Programy nabazgrane na świstku papieru dostają punkty karne już na starcie. Czasem bawię się w ich przepisywanie, lecz dużo chętniej sięgam po program nadesłany na kasecie czy dyskietce. Wiem wtedy, że się nie pomylę przy wklepywaniu i szybciej obejrzę końcowy rezultat. Najlepiej, gdy autor sam opisze swe dzieło i dołączy ten tekst również nagrany na nośniku. Tu jednak zaczyna się horror...

Niektórzy uważają, że nie warto się kłopotać polskimi literkami. Niestety, w gazecie to nie uchodzi. Więc siedzę nad tekstem, poprawiam,... Czasem przysnę, pozostawiam błędy. Jeden z AUTORóW przyjął kuriozalną metodę: "A" oznacza u niego "a", zaś "a" – ą. Zabawy z tym jest co niemiara! Niektórzy przeginają w drugą stronę. Otrzymałem ostatnio tekst w formacie IBM-owego edytora Chi-Writer. Łażę teraz po znajomych: pożycz stary Chi-Writera, bo mi czytelnik dogodził! Najchętniej witam teksty ASCII napisane na ATARI edytorem PE z Tajemnic lub PANTHER-em firmy AVALON. Jeżeli IBM, to też ASCII z polskimi literami w dowolnym standardzie (najchętniej Mazovia). Nośnik na życzenie odsyłamy.

No, to do roboty!

Redaktor

Tajemnice ATARI Pismo użytkowników ATARI XL/XE skr. pocz. 702, 35-959 Rzeszów 2
tel. (Rzeszów) 374-71 wewn. 250 lub 282. Redaktor naczelny: Janusz B. Wiśniewski
współpracownicy: Jarosław Syrylak, Jaromir Lechowicz
opracowanie merytoryczne: L. K. AVALON; opracowanie graficzne: Danuta Sienkowska
Wydawca: Tajemnice ATARI s.c. Adres redakcji: Rzeszów Targowa 1, pok. 801
Druk: PETIT. Indeks 379972

Prezentowany poniżej program jest rezydentnym zegarem przeznaczonym dla użytkowników DOS-u i COS-u. Po załadowaniu zegara możemy przystąpić do ustawienia aktualnego czasu.

Znaczenie klawiszy:

+, * wybór elementu do ustawiania
–, = zwiększanie, zmniejszanie wartości (w przypadku sekund - zerowanie)
CONTROL/+, CONTROL/* przemieszczanie zegara
RETURN, ESC koniec ustawiania

Od tej chwili Big Clock jest uczulony na klawisze konsoli. Oto, jak reaguje na ich naciśnięcie:

START/SELECT wyświetla aktualny czas (tak długo, jak klawisze są naciśnięte)
SELECT/OPTION pozwala na ponowne ustawienie czasu
START/SELECT/OPTION/RESET usuwa zegar z pamięci.

```
IJ  1000 REM *--------------------------*
SJ  1002 REM :    Big Clock ver 1.2     :
SK  1004 REM :        by Hedges         :
XY  1006 REM : (c) 1992 Tajemnice ATARI :
JP  1008 REM *--------------------------*
QJ  1010 DATA ffff008054483a201b50c9d2580
GT  1020 DATA bdc282950cca10f32040802080
QA  1030 DATA 80201582aec882acc9822042c6
BM  1040 DATA 6c0a00202480ad1fd0f020a200
TV  1050 DATA 8e4f83ca8e50832073800b00320
VC  1060 DATA 40804c8080a201bdd0829de702
AM  1070 DATA ca10f760a201bd4c819d2202bd
VU  1080 DATA 87819d2402bd258095 0cca10ec
ER  1090 DATA 207380b00ba201bdc0829de702
GQ  1100 DATA ca10f760ade702cdd082ade802
MB  1110 DATA edd18260a201bd22029d4c81bd
XD  1120 DATA 24029d8781ca10f1a906acc482
UR  1130 DATA aec582205ce4a907acc682aec7
GE  1140 DATA 824c5ce4a2018e4f83bd30029d
FJ  1150 DATA 518395cbbdcc829d5383ca10ef
CN  1160 DATA a000b1cb290fd00520 0f82d0f5
AB  1170 DATA 981865cb8d4483a5cc69008d45
OB  1180 DATA 83b1cb200f82c941f014c901d0
FR  1190 DATA f3b1cbaa200f82b1cb86cb85cc
SF  1200 DATA a000f0e32011 81a200bdce829d
WU  1210 DATA 3683bd538391cb200f829d3002
FW  1220 DATA e8e00290ea60a900aa9d5583e8
WE  1230 DATA e0dc90f860a201bd51839d5383
HR  1240 DATA 95cbca10f5a0008c4f83f0a7f8
XJ  1250 DATA a203bd4b831869019d4b83dd46
AE  1260 DATA 839008a9009d4b83ca10ead84c
VP  1270 DATA 4b81ad4f83d033a200b5cb48e8
AM  1280 DATA e00590f8ad1fd0c904d01320a7
QS  1290 DATA 80208981ad1fd0c904f0f6201d
EZ  1300 DATA 814c7e81c901d003201582a204
YK  1310 DATA 6895cbca10fa4c8681adce8285
DB  1320 DATA cbadcf8285cca91485ceac4a83
ER  1330 DATA a90285cfa6cfbd4b8320bd81c6
HV  1340 DATA cf3012a5cf48a98085cfa90a20
PE  1350 DATA ca816885cf4c9e816084cd4829
WC  1360 DATA 0f20ca81684a4a4a4a8dd1810a
AQ  1370 DATA 0a186900186dca828de481adcb
MA  1380 DATA 8269008de581a200bde38191cb
MI  1390 DATA 4820098268 91cb200982e8e005
EV  1400 DATA 90eca900ae5083e4cfd002a9ff
NA  1410 DATA 91cbc6cda4cd60981865cea860
UR  1420 DATA c8d002e6cc60a9008d508320a7
YP  1430 DATA 80208981adfc02c9fff0f6a2ff
AU  1440 DATA 8efc022083f9c91cf00ac90cf0
ED  1450 DATA 062046824c1d82201d81a9ff8d
XR  1460 DATA 50836 0a205ddf882f004ca10f8
AJ  1470 DATA 608a0aaabdd2828d6482bdd382
LW  1480 DATA 8d6582ae50834c6382ca1002a2
UP  1490 DATA 028e50836 0e8e00390f7a200f0
IR  1500 DATA f3e002f00dbd4b8318f86901d8
HO  1510 DATA dd46839002a9009d4b8360e002
ED  1520 DATA f0f6bd4b8338f8e901d810eebd
ZH  1530 DATA 468338f8e901d8d0e4ae4a83ca
ST  1540 DATA e00790038e4a834c1181ae4a83
JZ  1550 DATA e8e014b0f590f0000080248031
KB  1560 DATA 814e81e082fe82358355833184
BQ  1570 DATA 66826f8278828f82a782b58200
LH  1580 DATA 0042696720436c6f636b20312e
RA  1590 DATA 3220696e7374616c6c65649b06
VS  1600 DATA 070e0f86877e6666667e781818
NI  1610 DATA 187e7e067e607e7e061e067e66
BH  1620 DATA 667e06067e607e067e7e607e66
ZI  1630 DATA 7e7e0606060 67e667e667e7e66
ON  1640 DATA 7e067e0018001800 4b00000b0b
TC  1650 DATA 0b0b0b0b0b0b0b0b000b01000024
NX  1660 DATA 6060500700000000000000000000
TX  1670 DATA 0003184ed84ade7028de002ade8
SE  1680 DATA 028de102a90085cfa90085d2a9
YU  1690 DATA 8085d3ade70285d4ade80285d5
IU  1700 DATA 38a5d2e5d485d0a5d3e5d585d1
VP  1710 DATA a000f0033866cfa5d2c931a5d3
TS  1720 DATA e9849001602 4cf70613025b1d2
LV  1730 DATA 91d420e184aaf0e2c920f017c9
YV  1740 DATA 40f0ddc960f0d9290dc908f0d3
NE  1750 DATA 90418a291fc909f03ab1d2c8c9
HP  1760 DATA 00b1d288e9809012b1d2c8c932
PW* 1770 DATA b1d288e984b006a5d0a6d19007
AJ  1780 DATA 24cf30a5a900aa85d686d738b1
IM  1790 DATA d2e5d691d420e184b1d2e5d74c
ZM  1800 DATA d984b1d291d420e1844c6784e6
AE  1810 DATA d2d002e6d3e6d4d002e6d560e2
FJ  1820 DATA 02e3023184
```

(hed)

BASIC DOS

Program ten ułatwia współpracę ze stacją dysków podczas pracy z BASICem. Umożliwia wykonanie podstawowych operacji takich jak: zabezpieczenie i odbezpieczenie plików, skasowanie plików, zmiana nazwy plików, katalog dysku.

Forma wydawania rozkazów jest następująca:

`DOS:REM` *xxx filename*

gdzie *xxx* to trzyliterowy kod rozkazu, a *filename* to pełna nazwa pliku np:

`D:DOWOL.ASM`

Oto kody rozkazów:

- **LOK** - Zabezpieczenie pliku(ów)
- **UNL** - Odbezpieczenie pliku(ów)
- **ERA** - Skasowanie pliku(ów)
- **REN** - Zmiana nazwy - wymaga podania dwóch nazw plików oddzielonych od siebie przecinkiem (stara i nowa nazwa pliku)
- **DIR** - Katalog dysku

Dopuszczalne jest podanie w nazwie pliku znaków **?** i *****, o ile zezwala na to zastosowany DOS.

Program po wczytaniu relokuje się na granicę wolnej pamięci i podnosi ją następnie ponad siebie, aby zabezpieczyć się przed skasowaniem. Program jest "resetoodporny", a jego dezaktywacja następuje po wykonaniu "pustej" instrukcji DOS lub gdy format rozkazu jest niewłaściwy. Podczas dezaktywacji programu przywracane są: poprzedni wektor przerwania RESET i, o ile jest to możliwe, poprzednia granica pamięci. Granica pamięci nie zostaje obniżona w przypadku gdy przy aktywnym programie została ona zmieniona np. wczytany został XLFriend lub inny program modyfikujący wskaźnik MEMLO.

Dariusz Żołna

```
IJ 1000 REM *--------------------------*
NL 1002 REM : BASIC DOS                :
VB 1004 REM :  autor: Dariusz Zolna    :
YY 1006 REM : (c) 1992 Tajemnice ATARI :
JP 1008 REM *--------------------------*
NX 1010 DATA ffff00883c8aa202a5092901f0
KW 1020 DATA 02a20cb5008d2288b5018d2388
CQ 1030 DATA ad6289500ad638995014c2488
QN 1040 DATA 202188a50a8d8089a50b8d8189
US 1050 DATA ad648985 0aad658985 0bade702
GC 1060 DATA 8d8289ade8028d8389ad66898d
FI 1070 DATA e702ad67898de80260a002b18a
AC 1080 DATA c90cb03da202a5092901f002a2
GJ 1090 DATA 0cad22889500ad23889501ad80
ZB 1100 DATA 8985 0aad818985 0bade702cd66
SH 1110 DATA 89d014ade802cd6789d00cad82
SM 1120 DATA 898de702ad83898de8026c0a00
YV 1130 DATA a007b18ad0bda200a008b18add
TG 1140 DATA 6a89d013c8b18add6b89d00bc8
AI 1150 DATA b18add6c89d0034cc388e8e8e8
YR 1160 DATA e8e01190dc60e010b021bd6d89
GY 1170 DATA 8d7e89203c89ad7e899d4203a5
PR 1180 DATA 8a18690c9d4403a58b69009d45
NI 1190 DATA 034c56e4203c89a9039d4203a5
OV 1200 DATA 8a18690c9d4403a58b69009d45
GR 1210 DATA 03a9069d4a03a9009d4b032056
LW 1220 DATA e410034c5e89ae7f89a9009d48
BB 1230 DATA 038d48039d49038d4903a9059d
OE 1240 DATA 42032056e410034c5e89a20b8e
BW 1250 DATA 4203a2002056e44c1189a210bd
CA 1260 DATA 4003c9fff00a188a6910aae080
AW 1270 DATA 90f0608e7f8960ae7f89a90c9d
BB 1280 DATA 42034c56e44c53890021885188
LB 1290 DATA 848900004c4f4b23554e4c2445
AI 1300 DATA 52412152454e2044495206000 0
PQ 1310 DATA 00000000a90085cfa90085d2a9
QQ 1320 DATA 8885d3ade70285d48de002ade8
RA 1330 DATA 0285d58de10238a5d2e5d485d0
GS 1340 DATA a5d3e5d585d1a000f0033866cf
UK 1350 DATA a5d2c984a5d3e98990036ce702
QK 1360 DATA 24cf70613025b1d291d420308a
DA 1370 DATA aaf0e0c920f017c940f0dbc960
DY 1380 DATA f0d7290dc908f0d190418a291f
UT 1390 DATA c909f03ab1d2c8c900b1d288e9
SE 1400 DATA 889012b1d2c8c985b1d288e989
SP 1410 DATA b006a5d0a6d1900724cf30a3a9
XZ 1420 DATA 00aa85d686d738b1d2e5d691d4
OR 1430 DATA 20308ab1d2e5d74c288ab1d291
VE 1440 DATA d420308a4cb489e6d2d002e6d3
VM 1450 DATA e6d4d002e6d560e002e1028489
```

Wykaz punktów sprzedających bieżące i archiwalne numery Tajemnic Atari. Drukiem pogrubionym zaznaczono miejsca, w których można równocześnie nabyć legalne oprogramowanie firmy L.K. AVALON.

DRZAJA
Mirowska 6, Częstochowa

GM NEXUS
NMP 11, Częstochowa

SCALAK
NMP 39/41, Częstochowa

ELP Sklep Wielobranżowy
Targowisko, Pawilon IV/S
Gdańsk Wrzeszcz

MICROMAN
Pielęcytowa 31, Katowice
Rynek 4, Rybnik
DH HERMES, Rybnik
SUPERMARKET, Sosnowiec
Wojska Polskiego 14,
Bielsko-Biała

KRYMARYS
Toruńska 63, Koło

Księgarnia Techniczna
Podwale 4, Kraków

RETURN Studio Komputerowe
Zamojska 25, Lublin

KOBER, Grabowa 55a, Lublin

KSIĘGARNIA TECHNICZNA NOT
Komuny Paryskiej 5a, Łódź

#TRACK#
Reymonta 23, Opole
Kołłątaja 11, Opole

AR-WAL
Wyzwolenia/dego 1, Opole

P.H. METRO – bms
Ratajczaka 31, Poznań

Księgarnia Techniczna "DOM KSIĄŻKI"
Paderewskiego 6, Poznań

BAJT, Rejtana 33, Rzeszów

DABI, Szopena, Rzeszów

BARTEK, Kaźmierski, Rzeszów

DOMAR
Zygmuntowska, Rzeszów

STEL-DOM,
Pomorska 7, Słupsk

SONIX Sp. z o.o.
Rapackiego 3, Szczecin
DT Centrum, Szczecin
Rydla (Hallo 2), Szczecin

COMPUTER SHOP
Bol. Warszawy 18, Szczecinek

SKLEP WIELOBRANŻOWY
Żeromskiego 7, Suwałki

KWANT Agencja Handlowa
Żydowska 6, Tarnów

COMPUTER CENTER "WARS"
Marszałkowska 104/122
Warszawa

ATARI STUDIO
Generała Abrahama 4 Warszawa

O.K.W. ELEKTRONIKA
Mokotowska 51/53, Warszawa

ONKAR, Ignatka 26, Warszawa

DH ŁADA Sp
Oławska 27, Wrocław

DOMAIN SOFT
Kołłątaja 31, Wrocław

KSIĘGARNIA Staromiejska
Żeromskiego 16, Zielona Góra

GAME SOFT
Francuska 52, Zielona Góra

Zachęcamy właścicieli sklepów zainteresowanych sprzedażą naszego pisma lub oprogramowania, do kontaktu z naszą firmą. Najszybszy kontakt telefoniczny pod numer: Rzeszów 374-71 wew. 274, 275 w godz. 7.30 – 15:00. Posiadamy szeroki asortyment legalnych programów przeznaczonych dla różnych typów komputerów, oraz akcesoriów komputerowych. Oferujemy korzystne warunki współpracy. Firmy bądź sklepy zainteresowane kolportażem gazet otrzymają je na zasadach komisowych, a ponadto ich adresy będą co miesiąc drukowane w TA, co stanowi dobrą reklamę.

Oferujemy możliwość wysyłkowego zakupu niżej wymienionych akcesoriów komputerowych. Zamówienia (na kartach pocztowych) przyjmujemy pod adresem: L. K. AVALON skr. 46 38-100 STRZYŻÓW

Do podanych cen należy doliczyć koszt wysyłki (20.000 zł). W przypadku zamówienia kilku akcesoriów kwota ta doliczana jest do ich wartości tylko raz. Jeżeli zdecydują się Państwo zakupić u nas równocześnie z akcesoriami oprogramowanie z oferty umieszczonej na ostatniej stronie pisma, koszt wysyłki nie jest doliczany do wartości zamówionego towaru. Zamówienia realizujemy drogą pocztową, należność uiszczana jest przy odbiorze przesyłki. Wszystkie ceny podane są w zł i ważne do ukazania się następnego numeru Tajemnic Atari.

Dyskietki 5.25'

zamawiana ilość	10	20-40	50-100
	cena za sztukę		
No name DD	7200	6700	6300
VERBATIM COLORS DD	12200	11400	10700
BASF DD	13000	12000	11300
VERBATIM datalife + DD	15000	14000	13000

Joysticki

QUICKJOY II (auto fire, styki blaszane)	115000
QUICKSHOT II (auto fire, styki blaszane)	115000
QUICKSHOT II plus (auto fire, mikrostyki)	160000

Dyskietki czyszczące głowice w stacjach dysków

5.25'	50000
3.5'	50000
Kaseta czyszcząco-rozmagnesowująca do magnetofonów	25000

Filtry monitorowe siatkowe

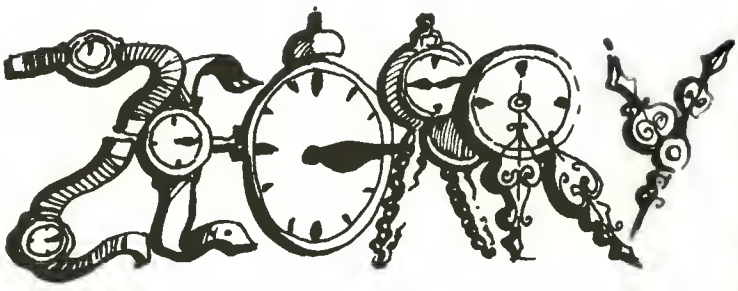
monochromatyczny, zachodni, 14 cali	150000
mono/kolor, 14 cali, zachodni	190000
CONCETTO (mono/kolor, 14 cali, zachodni)	215000
CONCETTO (mono/kolor, 14 cali, z uziemieniem, zachodni)	250000

Ponadto

"złodziejka antenowa" - umożliwia podłączenie do telewizora anteny i komputera jednocześnie	35000
Magic Clip - uchwyt na fiszki do monitora	35000
Dziurkacz do dyskietek (zezwolenie na zapis)	50000
Pokrywy na komputery Atari 65/130/800 XE	50000
Taśma barwiąca do drukarek STAR LC 10/20	70000
Złącze MICROPRINT - umożliwiające podłączenie do Atari dowolnej drukarki -	190000

Literatura

Leksykon komputerowy dla początkujących (72 str.)	25000
Słownik terminów i komunikatów komp. (72 str.)	25000
Instrukcja Atari 65 XE/130 XE (128 str.)	30000
Atari Basic dla dzieci (102 str.)	30000
Logo Atari	30000
Poznajemy komputer Atari XL/XE (190 str.)	30000
Podręcznik Atari XL/XE (110 str.)	30000
Język maszynowy Atari XL/XE (156 str.)	30000
Mac 65 (86 str.)	30000
Bug 65 (42 str.)	30000
Opisy gier na małe ATARI	35000
Języki Atari XL/XE (238 str.)	35000
Mapa pamięci Atari XL/XE,	
Dyskowe systemy operacyjne (188 str.)	40000
Poradnik użytkownika Atari XL/XE (178 str.)	40000
Podstawy programowania w ATARI basic (170 str.)	45000
Poradnik Programisty Atari (236 str.)	50000
Atari basic	55000
Atari logo (288 str.)	55000



Dziś rewia zegarów. Ten szczególnie umiłowany przez Was temat był przez nas traktowany dotąd po macoszemu. Z banalnej przyczyny: choć programów tego typu przysyłacie wiele, tylko nieliczne są na tyle dobre, by je można było bez wstydu komuś pokazać.

Główny grzech – to zabijanie DOS-u. Z niewiadomych powodów użytkownicy magnetofonów nienawidzą DOS-u i zabijają go przez

```
POKE 12,L:POKE 13,H:POKE 9,1
```

W praktyce każdy program, w którym widzę tę magiczną sekwencję rozkazów, leci natychmiast do kosza.

Grzech drugi: wyświetlanie. Nadsyłane zegary przeważnie zasłaniają coś na ekranie lub same są przez coś zamazywane, mrugają podczas różnych operacji, ulegają metamorfozie w różnych trybach graficznych.

Grzech trzeci: przeciążanie przerwań. Większość zegarów ma ambicję nigdy nie zamierać. Korzystają więc z bardzo mocnych przerwań – VBLKI lub DLI. Nieumiejętne ich użycie powoduje zakłócenia komunikacji z szybkimi urządzeniami WE/WY, np. ze stacją dysków.

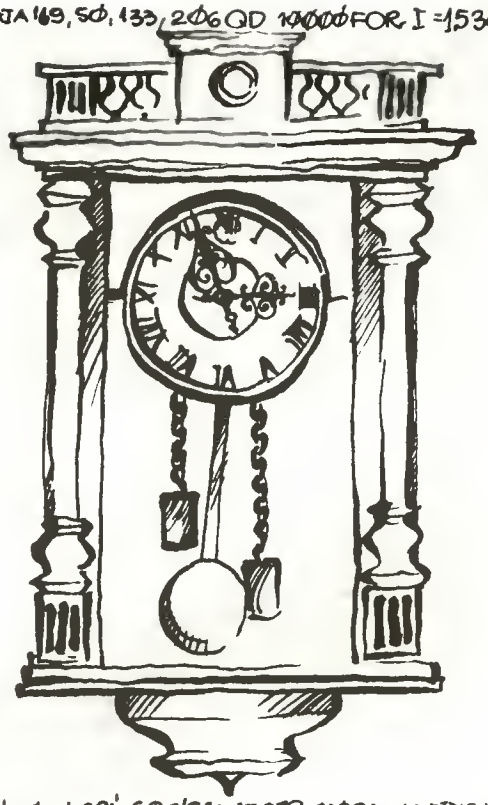
Dziś prezentujemy garść czasomierzy nie dlatego, że są one całkiem wolne od usterek, lecz w nadziei, że wpłyną na postawę ich przyszłych (lub niedoszłych) autorów.

1. **Bargarian**: Program modyfikuje istniejącą DL, dopisując kilka bajtów z przodu. Ponieważ jednak nie zmienia przy tym wskaźnika MEMHI, może powodować konflikty z innymi programami. Zegar znika po wykonaniu rozkazu GRAPHICS, a pojawia się znów po RESET. To częsty błąd! Program na 6. stronie powinien umrzeć po RESET, robiąc miejsce dla swego następcy.

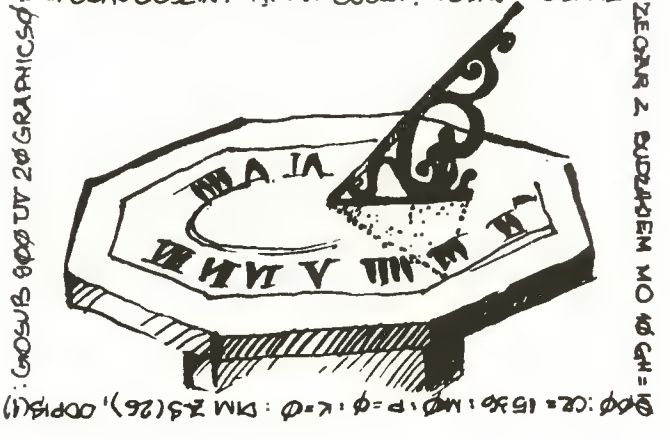
2. **Marcin Kania**: Zegar ustawia się przez SELECT/OPTION, START/SELECT i START/SELECT/OPTION Wyświetla się na zastanym ekranie (także w trybach innych niż 0, lecz wtedy trudniej go odeczytać), w dodatku w stałym miejscu pamięci, zamiast we wskazanym przez słowo 88. Zamazuje część ekranu, co jest nieraz dość kłopotliwe. Na szczęście RESET go zabija.

3. **Jacek Binkul**: Niszczy łańcuch procedur inicjalizacyjnych wskazywany przez słowo 12. Oczywiście blokuje przy tym nieodwracalnie szóstą stronę. Można tego uniknąć, wyrzucając "poki" z wiersza 90.

```
NG  1 REM
WY  2 REM ZEGAR  by  BARGARIAN
KR  3 REM COPYRAIGHT BY TAJEMNICE ATARI
NJ  4 REM
XB  9 N=1536
IL  10 READ K:IF K<0 THEN GOTO 12
NO  11 POKE N,K:N=N+1:GOTO 10
XP  12 ? CHR$(125)
XF  13 ? "Podaj ktora jest godzina ";:INPU
    T GOD
EG  14 ? "--------|-------  minuta ";:INPU
    T MIN
DE  15 ? "--------|------- sekunda ";:INPU
    T SEK
BK  16 REM
BN  24 POKE 1784,INT(GOD/10)*16+(GOD-INT(G
    OD/10)*10)
KV  26 POKE 1785,INT(MIN/10)*16+(MIN-INT(M
    IN/10)*10)
GX  27 POKE 1786,INT(SEK/10)*16+(SEK-INT(S
    EK/10)*10)
BP  28 REM
WF  29 X=USR(1536)
WY  30 DATA 104,165,12,141,24,6,165,13,141
    ,25,6,32,27,6,76,92,228,32,27,6,32,92,
    228,32,32
JG  31 DATA 78,96,173,49,2,133,204,173,48,
    2,56,233,3,141,48,2,133,203,160,0,185,
    201,6,145,203
VE  32 DATA 200,192,6,208,246,160,33,173,4
    8,2,145,203,169,6,160,69,162,6,96,169,
    1,133,9,169,17
BP  33 DATA 133,12,169,6,133,13,206,251,6,
    208,67,169,50,141,251,6,248,173,250,6,
    24,105,1,141,250
KL  34 DATA 6,201,96,208,14,169,0,141,250,
    6,173,249,6,24,105,1,141,249,6,173,249
    ,6,201,96,208
BQ  35 DATA 14,169,0,141,249,6,173,248,6,2
    4,105,1,141,248,6,173,248,6,201,36,208
    ,5,169,0,141
UJ  36 DATA 248,6,216,216,160,0,173,248,6,
    32,177,6,173,249,6,32,177,6,173,250,6,
    32,177,6,76
QU  37 DATA 95,228,170,74,74,74,74,24,105,
    16,153,207,6,200,138,41,15,24,105,16,1
    53,207,6,200,200
YE  38 DATA 96,112,64,66,207,6,32,0,0,26,0
    ,0,26,0,0,0,0,0,0,0,0,0,0,0,0
KQ  39 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,35,89,80,50,-1
```



```
QD 10000 FOR I=1536 TO 1788:READ A:POKE I
   ,A:NEXT I:I=USR(1779):POKE 206,50:RETU
   RN
WI 10010 DATA 198,205,165,205,201,0,208,4
   ,169,10,133,205,198,206,165,206,240,3,
   76,169,6
FK 10020 DATA 169,50,133,206,238,141,1,16
   9,10,205,141,1,208,100,169,0,141,141,1
   ,238,140
CN 10030 DATA 1,169,6,205,140,1,208,85,16
   9,0,141,140,1,238,138,1,169,10,205,138
   ,1
BU 10040 DATA 208,70,169,0,141,138,1,238,
   137,1,169,6,205,137,1,208,55,169,0,141
   ,137
MI 10050 DATA 1,173,142,1,201,1,240,43,23
   8,135,1,169,4,205,135,1,240,18,169,10,
   205
OC 10060 DATA 135,1,208,26,169,0,141,135,
   1,238,134,1,76,135,6,169,2,205,134,1,2
   08
TI 10070 DATA 8,169,0,141,135,1,141,134,1
   ,173,143,1,201,0,208,24,160,0,24,185,1
   34
LL 10080 DATA 1,105,16,153,64,156,200,192
   ,8,208,242,169,26,141,66,156,141,69,15
   6,76,98
DW 10090 DATA 228,173,31,208,201,0,208,6,
   141,141,1,141,140,1,174,142,1,240,6,16
   6,205
SX 10100 DATA 224,10,208,198,162,1,142,14
   2,1,201,1,208,3,76,92,6,201,4,208,3,76
WS 10110 DATA 55,6,206,142,1,173,143,1,24
   0,171,201,2,240,167,238,143,1,160,0,18
   5,124
KR 10120 DATA 1,153,64,156,200,192,8,208,
   245,76,135,6,104,160,0,162,6,169,7,76,
   92,228
```



```
TB 2 REM ZEGAR Z BUDZIKIEM
HO 10 GH=1000:CZ=1536:M=0:P=0:K=0:DIM A$(
   26),ODP$(1):GOSUB 900
UV 20 GRAPHICS 0:? "PODAJ GODZINY ":INPUT
    GODZ:? "PODAJ MINUTY ":INPUT MIN:? "P
   ODAJ SEKUNDY ":INPUT SEC
BU 30 IF GODZ>23 OR MIN>59 OR SEC>59 THEN
   GOSUB 100:GOTO 20
VI 40 POKE 1773,INT(GODZ/10):POKE 1772,GO
   DZ-INT(GODZ/10)*10:POKE 1770,INT(MIN/1
   0):POKE 1769,MIN-INT(MIN/10)*10
CJ 50 POKE 1767,INT(SEC/10):POKE 1766,SEC
   -INT(SEC/10)*10
KB 60 ? "USTAWIC BUDZIK [T/N] ":INPUT ODP
   $:IF ODP$="N" THEN POKE 1689,96:GOTO 9
   0
BU 70 ? "PODAJ GODZINY ":INPUT GODZ:? "PO
   DAJ MINUTY ":INPUT MIN:IF GODZ>23 OR M
   IN>59 THEN SEC=0:GOSUB 100:GOTO 70
BV 80 POKE 1777,INT(GODZ/10):POKE 1776,GO
   DZ-INT(GODZ/10)*10:POKE 1775,INT(MIN/1
   0):POKE 1774,MIN-INT(MIN/10)*10
JH 90 POKE 9,1:POKE 12,1:POKE 13,6:I=USR(
   1536):END
VL 100 CLOSE #1:? "ZLE PODALES CZAS - ";G
   ODZ;":";MIN;":";SEC:? :OPEN #1,4,0,"K:
   ":GET #1,A:CLOSE #1:RETURN
HT 900 RESTORE GH:FOR I=1 TO 19:READ A$
WN 910 FOR J=1 TO LEN(A$) STEP 2
OW 920 M=ASC(A$(J))-48:M=(M-7*(M>9))*16:P
   =ASC(A$(J+1))-48:P=P-7*(P>9):POKE CZ+K
   ,M+P:K=K+1:NEXT J:NEXT I
ZL 930 RETURN
KE 1000 REM ZEGAR
DG 1010 DATA 68A9008DF306A001A200A90220
CL 1020 DATA 5CE4A018A206A9094C5CE4EEE6
MD 1030 DATA 06ADE606C90A9051A9008DE606
VI 1040 DATA EEE706ADE706C9069042A9008D
WL 1050 DATA E706EEE906ADE906C90A9033A9
HG 1060 DATA 008DE906EEEA06ADEA06C90690
RD 1070 DATA 24A9008DEA06EEEC06ADEC06CD
HU 1080 DATA F2069014A9008DEC06EEED06AD
GG 1090 DATA ED06C9039005A9008DED06A90A
XA 1100 DATA 8DF206ADED06C9029005A9048D
VA 1110 DATA F206A9328D1A02A020A207BDE6
BC 1120 DATA 0618699095158C8CA10F4ADED06
GG 1130 DATA CDF106D01DADEC06CDF006D015
QD 1140 DATA ADEA06CDEF06D00DADE906CDEE
NZ 1150 DATA 06D005A9028DF306ADF306C900
LA 1160 DATA F012CEF406F00EA92F8D02D28D
MO 1170 DATA C802A9C88D63D260A9008D03D2
XB 1180 DATA 8DC802A9028DF4066000000A00
IU 1190 DATA 000A0000000000000A0002
```

Słowo "BATCH" wywodzi się z ery prehistorycznych komputerów, które zajmowały wielkie, klimatyzowane hale i wymagały całego sztabu ludzi do obsługi. W tych czasach zwykły śmiertelnik nie był dopuszczany w pobliże dostojnej maszyny – zadania do wykonania dostarczało się na kartach perforowanych. Taki plik kart opisujący kolejne polecenia użytkownika dla systemu nazywa się właśnie **wsadem** (po angielsku: batch). Ten sposób pracy z komputerem nazywa się przetwarzaniem **wsadowym**, w przeciwieństwie do pracy interakcyjnej, polegającej na bezpośrednim kontakcie człowieka z komputerem poprzez zestaw klawiatura/monitor. Raz uruchomiony **wsad** wykonuje żądane czynności bez dalszej ingerencji operatora.

Zaletą przetwarzania wsadowego jest to, że można uruchamiać te same lub podobne sekwencje poleceń korzystając ze stałego, raz przygotowanego ich zestawu. Typowym przykładem jest przenoszenie plików do RAM-dysku, np. przy pracy z Quick Assemblerem.

Powyższy listing należy połączyć ze Zgrywusem+ (TA 1/92). Po uruchomieniu zgrać go jako plik, który odtąd będzie stałym nagłówkiem wszystkich wsadów. Aby otrzymać gotowy "batch", trzeba przygotować jego tekst (dowolnym edytorem), a następnie połączyć go z MINI-BATCHem za pomocą programu APPEND (TA 5/91).

```
IJ  1000 REM *--------------------------*
YC  1002 REM : MINI-BATCH                :
DT  1004 REM :  autor: J.B. Wi$niewski   :
YI  1006 REM : (c) 1992 Tajemnice ATARI :
JP  1008 REM *--------------------------*
UV  1010 DATA ffff0090859Ø27Ø bd4003300 7
BO  1020 DATA bd4203c907f00a388ae910aad0
ZB  1030 DATA ed6c0a00a9299d4403a9069d45
JD  1040 DATA 03ad0a069d4803a9009d4903 20
BP  1050 DATA 56e4bc4803 8c0a06f04da99b99
IN  1060 DATA 2806a9009d4803 2056e410fbad
ET  1070 DATA 2103 85d48d0e06ad2203 85d58d
VL  1080 DATA 1306a00bb1d4991d06881 0f8ae
NC  1090 DATA 2306ac2406e8d001c88e1906 8c
QT  1100 DATA 1a06a9ff8d2106a9058d2206a9
VN  1110 DATA 1d8d2103a9068d2203600006 1c
HT  1120 DATA 06a000b92906c88c0106c0d790
QK  1130 DATA 0aa0008c2103a0008c2203482 0
UR  1140 DATA 00006860e002e3021c060090
```

Program tego typu był już prezentowany w TA 1/92. MINI-BATCH pokazuje inne podejście do tego zagadnienia. Cały program mieści się na szóstej stronie, wliczając w to 200-bajtowy bufor tekstu. Z doświadczenia wynika, że taka długość tekstu jest przeważnie wystarczająca.

Janusz B. Wiśniewski

MASTER HEAD

Kilka lat temu użytkowników komputerów IBM ogarnął szał. Każdy, kto tylko zdołał kupić (lub skopiować od kolegi) grę pod tytułem "Sokoban", spędzał całe godziny próbując ją przejść i zobaczyć upragniony napis CONGRATULATION.

Co sprawiało, że gra posiadająca proste zasady i niezbyt zachwycającą grafikę, potrafiła być aż tak wciągająca? Tym czymś była wymagana od grającego konieczność logicznego myślenia.

Zasady gry, jak już wspomnieliśmy, były bardzo proste: ludzik poruszający się po labiryncie (zwanym magazynem) musiał poprzesuwać znajdujące się w nim paczki na wyznaczone miejsce. Należało zrobić to dokładnie i ostrożnie, bowiem każdy nierozważny ruch mógł uniemożliwić wykonanie zadania i zniweczyć całą dotychczasową pracę.

O tym, jak wciągająca jest ta gra, można się przekonać, próbując samemu w nią zagrać. Jest to możliwe za sprawą programu o nieco przewrotnym tytule "Master Head". Gra ta powstała w 1992 roku i różni się nieco od swojego pierwowzoru. Ma ona dobrze opracowaną i wykonaną szatę graficzną oraz muzyczną. Oprawę dźwiękową do gry napisał Tomasz Liebich, który jest obecnie jednym z najlepszych twórców muzyki na 8-bitowe komputery Atari. Gra zawiera kilkanaście minut wspaniałej muzyki napisanej z użyciem tzw. digitaliz.

Gra "Master Head" składa się z 46 labiryntów, na ułożenie których grający posiada niezliczoną ilość czasu. Gracz może posługiwać się joystickiem lub klawiaturą. Dodatkowe informacje o grze i funkcjach niektórych klawiszy można uzyskać po naciśnięciu klawisza HELP.

SKARBNIK

Gra logiczna. W dwóch studniach znajdują się klejnoty. W lewej studni są one ułożone w sposób przypadkowy (tak zastaje je skarbnik), w prawej natomiast widzimy do jakiego ustawienia mamy doprowadzić. Klejnoty porządkujemy zgodnie z pewnymi, opisanymi w instrukcji zasadami. Gra mimo swej prostoty bardzo wciąga – nie mniej niż legendarny już ROBBO. Niezaprzeczalnymi walorami programu są ponadto ładna grafika i ciekawa muzyka.

I.Q. MASTER

Gra zręcznościowo-logiczno-edukacyjna, podobna nieco do nieśmiertelnego Tetrisa. Do studni wpadają klocki z literkami. Zadaniem grającego jest ułożenie spadających klocków w wyrazy, które podane są z boku studni. W momencie ułożenia jednego z podanych wyrazów pojawia się jego słownikowe wyjaśnienie (w przypadku trudniejszych wyrazów), a ze studni znika cała linia, w której znajdował się ułożony wyraz. Wyrazy są dobrane w sposób umożliwiający graczowi poznanie nowych, ciekawych słów. Do gry dołączony jest prosty słownik dla dzieci oraz słownik wyrazów obcych. Istnieje możliwość dogrywania dalszych słowników. W przygotowaniu znajduje się program uczący słówek angielskich oraz dalsze rozszerzenie słownika wyrazów obcych.

Grafika gry wykorzystuje maksymalną rozdzielczość Atari, co wcale nie oznacza, że jest dwukolorowa. Jak zawsze w produktach firmy ASF zwraca uwagę interesująca muzyka.

Gra przygotowywana jest również w wersji dla komputerów IBM i AMIGA.

Łata do ACTION!

Kolejną pożyteczną "łatę" przysłał pan Adam Salamon. Jego poprawka dotyczy języka ACTION!. Autor pisze:

Rzeczą dosyć uciążliwą (dla mnie) jest ustawianie parametrów pracy poprzez tzw. menu opcji. Czynność tę należy wykonać po wczytaniu ACTION!, a także po wykonaniu komendy BOOT z monitora, gdyż ACTION! przywraca standardowe parametry.

Wpisując poprawkę można tak "przerobić" ACTION!, by po wczytaniu uruchamiał się z dowolnie ustawionymi parametramI DISPLAY, BELL, CASE SENSITIVE, LINE SIZE, WINDOW 1 SIZE i EOL CHAR. Parametry te będzie można oczywiście zmieniać korzystając z menu opcji, ale po wykonaniu komendy BOOT zostaną one i tak ustawione w wybrany na początku sposób.

```
LA 1000 REM DISPLAY - 0-N 34-Y
CT 1010 DATA 973,34,0
IC 1020 REM
AQ 1030 REM BELL - 96-N 76-Y
AX 1040 DATA 956,76,96
IL 1050 REM
JI 1060 REM CASE SENSITIVE - 223-N 255-Y
SF 1070 DATA 934,223,255
IU 1080 REM
WV 1090 REM WINDOW 1 SIZE
UH 1100 DATA 938,18,18
IB 1110 REM
CN 1120 REM LINE SIZE
KS 1130 DATA 939,120,120
IK 1140 REM
ST 1150 REM EOL CHAR - ZNAK MUSI BYC PODA
   NY W KODZIE INTERNAL
UF 1160 DATA 959,0,0
```

Za pomocą mojej poprawki nie można zmienić ustawienia parametrów LIST, TRACE i LEFT MARGIN. Zmiana pierwszych dwóch jest dosyć trudna, gdyż są one w inny sposób zapisywane przez ACTION! oraz mija się to z celem (nie istnieje chyba potrzeba, by po wczytaniu opcja LIST lub TRACE była włączona). Parametr LEFT MARGIN różni się w znaczny sposób od innych. Jest tak dlatego, że ACTION! operuje bezpośrednio na komórce 82, czyli komenda BOOT nie przywraca jego poprzedniego ustawienia. Jeżeli ktoś koniecznie chce, aby po uruchomieniu ACTION! lewy margines był inny niż 2, może przepisać poniższy program pod QUICK ASSEMBLEREM:

```
opt 37
org 82
dta b(0) - zamiast 0 może być inna liczba
end
```

Po wygenerowaniu przez QA kodu należy go połączyć z ACTION! za pomocą programu APPEND z TA 5/91 (najpierw kod, potem ACTION!).

Przedstawioną w postaci wierszy DATA "łatę" należy połączyć z programem PATCHER z TA 7/91 i całość wykonać. To pozwoli na wczytanie pliku ACTION! i zapisanie jego poprawionej wersji.

Ludzie listy piszą...

Pan Dawid Weiss z Piły pisze:

W Quick Assemblerze (wersja kasetowa) wykryłem drobny błąd (zamierzony?). Korzystam z Turbo 2000 i QA nie chciał nagrywać programu na taśmę. W procedurze obsługi zapisującej plik pod hasłem CMP #'D' następnym jest zły adres skoku. Program poprawiłem, lecz chciałbym się doczekać odpowiedzi, dlaczego tak jest?

Drogi Panie Dawidzie, podziwiam Pańską dociekliwość i umiejętność analizy programu maszynowego. Istotnie, QA w procedurze zapisu, po wykryciu, że urządzeniem wyjściowym jest "D:" przechodzi do otwarcia pliku w trybie odczytu. Nie jest to jednak błąd, lecz metoda sprawdzenia, czy taki plik już istnieje na dysku. Nazwa urządzenia "D:" bowiem jest zgodnie z tradycją zarezerwowana dla stacji dysków, a Turbo 2000 niefrasobliwie jej używa. QA natychmiast po otwarciu pliku znów go zamyka, po czym dopiero przystąpi do zapisu. Dlatego posiadacze systemu Turbo 2000 przy próbie zapisu pliku ze zdziwieniem słyszą sygnał do odczytu. Rozwiązanie tego problemu jest proste: należy zmienić nazwę urządzenia instalowanego przez Turbo, np. na "T:". Było to opisane w "łatach" prezentowanych na łamach TA.

Janusz B. Wiśniewski

tajemnice ATARI
skr. poczt. 702
35-959 Rzeszów 2

Zamawiam nagrane na ☐ kasecie ☐ dysku
programy zamieszczone w ------- numerze TA

mój adres:
_____
_____
_____

PrenumeraTA

Liczba egzemplarzy każdego numeru	cena 1 egzemplarza	3-mies.	6-mies.
n=1	8 900	26 700	53 400
n=2	8 300	49 800	99 600
n=3	8 000	72 000	144 000
n=4	7 650	91 800	183 800
n=5 do 9	7 300	21 900*n	43 800*n
n=10 lub więcej	6 750	20 250*n	40 500*n

W celu zamówienia prenumeraty należy wypełnić czytelnie blankiet wpłaty na konto (dostępny na pocztach i w bankach) podając na odwrocie imię, nazwisko i adres zamawiającego oraz liczbę egzemplarzy danego numeru wraz z okresem prenumeraty. Pieniądze prosimy wpłacać na konto:

"Tajemnice Atari" s.c. Rzeszów, ul. Targowa 1/801
Bank Przemysłowo-Handlowy III Oddział Rzeszów
nr 326807-23650-136

Redakcja **Tajemnic ATARI** informuje, że posiada jeszcze dużą liczbę egzemplarzy archiwalnych czasopisma. Można je nabywać wysyłkowo wpłacając pieniądze podobnie, jak przy prenumeracie (przekazem bankowym na konto podane wyżej). Oto wykaz posiadanych przez nas numerów wraz z cenami:

TA 5/91	**Cena: 6 900 zł**
TA 6/91	**Cena: 6 900 zł**
TA 7/91	**Cena: 6 900 zł**
TA 8/91	**Cena: 8 900 zł**
TA 1/92	**Cena: 8 900 zł**

Polecam zestaw rewelacyjnych programów kopiujących, odbezpieczających i zabezpieczających (wersja dyskowa i kasetowa). Cena 25 000 złotych + koszty przesyłki

Dominik Kulawik
ul. Tuwima 14a
34-500 Zakopane
tel. 46-15

USŁUGI KOMPUTEROWE

Naprawy sprzętu

- stacje dysków
- komputery
- joysticki
- magnetofony
- monitory

Modernizacje

- rozszerzenia pamięci
- TURBO do stacji dysków i magnetofonów

Oprogramowanie

Rzeszów, ul. Poznańska 2a tel. 447-20 w. 9

Ceny ogłoszeń ramkowych w TA:

• 1 cm^2	12000
• Cała strona	6 mln. + 100% za kolor
• Ostatnia strona (kolor)	15 mln.

Ogłoszenia drobne:

• każde słowo	5 000

"PATROL" – nowa gra strategiczna oraz edytor plansz. Cena (dysk/kaseta) 28 000 + koszty przesyłki.
MACIEJ ŚLIFIRCZYK, ul. Grójecka 64/1
02-339 Warszawa

☐ **Księga przychodów i rozchodów – dysk+instrukcja, tabulogramy. Cena 70000 zł**

☐ **Test inteligencji – dysk, kaseta. Cena 30000 zł**

☐ **Kabała - rozrywka parapsychologiczna. Cena 30000 zł**

oraz inne programy użytkowe
Katalogi gratis

Warszawa DH Sezam Studio Zet
ul. Marszałkowska 126
UPT 66 skr. poczt. 14

Redakcja **Tajemnic Atari** informuje o możliwości wysyłkowego nabywania wszystkich programów zawartych w dowolnym numerze pisma, nagranych na kasetę (tylko w transmisji normalnej) lub na dyskietkę. W skład zbioru wchodzą wszystkie listingi, zamieszczone w wybranym przez Zamawiającego numerze Tajemnic.

Cena wersji kasetowej wynosi 25.000 zł, zaś dyskowej 23.000 zł. W cenę wliczony jest koszt nośnika, nagrania, opakowania oraz opłaty pocztowe. Same listingi są za darmo.
Zainteresowanych ofertą prosimy o dokładne wypełnienie zamówienia (na odwrocie) i po naklejeniu go na kartę pocztową, wysłanie na adres redakcji podany w stopce.

Uwaga! Prosimy o nieprzysyłanie wcześniej należności, będzie ona bowiem regulowana przy odbiorze przesyłki.

tu jest miejsce na wklejenie Twojego ogłoszenia

REDAKCJA NIE PONOSI ODPOWIEDZIALNOŚCI ZA TREŚĆ ZAMIESZCZANYCH OGŁOSZEŃ.

MAPA PAMIĘCI

562, $0232 SSKCTL

Rejestr sterujący portu szeregowego, cień komórki o adresie 53775 ($d20f). Ponieważ korzystanie z tego rejestru ma sens tylko w trybie zapisu, podam jakie efekty wywołuje ustawienie poszczególnych bitów w tej komórce:

0 Włącza układ eliminujący drgania klawiatury.
1 Włącza układ odczytu klawiatury.
2 Zwiększa częstotliwość odczytu potencjometrów. Zamiast co jeden VBLANK, pomiar będzie się odbywał co dwie linie obrazu.
3 Włącza dwutonową, zamiast zero-jedynkowej, transmisję szeregową.
4-6 Odpowiednia kombincja służy do wyboru trybu pracy portu szeregowego.
7 Wymuszenie przerwania; wyzerowanie wyjścia portu szeregowego.

Standardowo system wpisuje tutaj 19, czyli $13.

563, $0233 SPARE

Pojedynczy bajt, nie używany przez system operacyjny. Firma ATARI nie zaleca jednak korzystania we własnych programach z żadnej spośród tego typu komórek, przewidując ewentualne wykorzystanie ich w późniejszych wersjach systemu.

Tak się składa, że opis kolejnych dwóch komórek pamięci sprawia mi pewne trudności, wynikające z faktu wykorzystania przez hardware ATARI lokacji 564 i 565 ($0234, $0235) do przechowywania pozycji poziomej i pionowej pióra świetlnego. Niestety, nie zetknąłem się jeszcze bezpośrednio z piórem świetlnym, przeznaczonym dla naszego komputera, stąd opieram się jedynie na książce "Mapping the ATARI" I. Chadwicka.

564, $0234 LPENH

Pozioma pozycja pióra świetlnego, liczona w cyklach koloru. Zakres wartości od 0 do 227.

565, $0235 LPENV

Pionowa pozycja pióra świetlnego. Zakres wartości od 0 do 95.

Zawartość obu tych komórek jest modyfikowana za każdym razem, gdy użytkownik naciska guzik na piórze świetlnym.

Pełny opis współpracy komputera z piórem można znaleźć w książce "Hardware Manual" na stronie II-32.

566, 567, $0236, $0237 BRKKY

Wektor przerwania wywoływanego naciśnięciem klawisza BREAK. Standardowo ustawiany na 59220 ($e754). Zmiana tego wektora umożliwia dołączenie do systemu własnej procedury obsługi klawisza BREAK.

568, 569, $0238, $0239 ...

Są to dwie komórki pamięci, które nie mają oficjalnej etykiety i nie są używane przez system operacyjny komputera.

W tym miejscu zaczyna się czterobajtowy bufor komendy używany przez SIO podczas transmisji szeregowej. Użytkownik nie powinien modyfikować zawartości tych komórek, ponieważ może to spowodować nieprzewidziane efekty.

570, $023a CDEVIC

Numer identyfikacyjny szyny, używany przez SIO.

571, $023b CCOMND

Kod rozkazu szyny, ustawiany przez SIO.

572, $023c CAUX1

Bajt pomocniczy dla komendy, przenoszony z lokacji 778 ($030a).

573, $023d CAUX2

Bajt pomocniczy dla komendy, przenoszony z lokacji 779 ($030b).

I jeszcze kilka innych komórek używanych przez procedury transmisji szeregowej.

574, $023e TEMP

Rejestr pomocniczy, używany przez SIO.

575, $023f ERRFLG

Flaga błędu transmisji szeregowej.

576, $0240 DFLAGS

Flaga dysku, odczytana z pierwszego bajtu pierwszego sektora dysku.

577, $0241 DBSECT

Ilość tzw. "boot sectors", czyli sektorów zawierających kod procedury odczytu potrzebnych po włączeniu komputera plików. Odczytana z pierwszego sektora dysku.

578, 579, $0242, $0243 BOOTAD

Adres ładowania kodu, przeprowadzającego odczyt potrzebnych po włączeniu komputera plików. Ten sam adres bywa także chwilowo wpisywany do komórki RAMLO (4, 5). Po odczytaniu, zawartość "boot sectors" zostaje przeniesiona pod podany tutaj adres. Normalnie, np. DOS-y ładują ten swój fragment od adresu 1792 ($0700)

580, $0244 COLDST

Flaga zimnego startu. W zależności od tego, jaka wartość jest wpisana do tej komórki system operacyjny komputera w różny sposób reaguje na naciśnięcie klawisza RESET. Normalnie ta komórka pamięci zawiera zero, dzięki czemu użycie klawisza RESET nie powoduje restartu systemu, połączonego z próbą odczytu wstępnego. Wpisanie do COLDST dowolnej wartości niezerowej zmusza system operacyjny do wykonania wszystkich akcji, zazwyczaj wykonywanych zaraz po włączeniu zasilania.

Posiadaczom pakietu QA proponuję podejrzenie zawartości tej komórki pamięci przy pomocy programu XLF w sytuacji normalnej pracy i po wywołaniu komendy BYE interpretera BASIC-u. Wcale nie musimy zgodzić się na zimny start komputera! Wystarczy wyzerować COLDST oraz zmienić zawartość komórki 202 (to wymagania BASIC-u), aby powrócić do przerwanej pracy.

581, $0245 ...

Kolejna komórka bez przydziału.

582, $0246 DSKTIM

Rejestr zawierający tzw. timeout dla operacji dyskowych. Procedury komunikacji ze stacją dysków zapisują tutaj maksymalny czas, w którym napęd musi zakończyć zleconą mu operację. Standardowo OS wpisuje tutaj wartość 224, co ma oznaczać, że sterownik dysku musi zakończyć pracę w ciągu siedmiu sekund od otrzymania polecenia. Wystarcza to do wykonania każdej operacji za wyjątkiem formatowania dyskietki, stąd programy, które to robią, same ustawiają odpowiednio dłuższy czas.

583-622, $0247-$026e LINBUF

Czterdziestoznakowy bufor wiersza. Używany przez edytor do chwilowego przechowywania tekstu podczas przesuwania ekranu. Procedura przesuwająca zapisuje wektor tego bufora w komórkach 100 i 101 ($64, $65).

623, $026f GPRIOR

Rejestr wyboru priorytetu, cień komórki 53275 ($d01b). Używany głównie do określenia, które z obiektów ekranowych mają być wysunięte przed inne, do tworzenia piątego gracza oraz do wybierania koloru nakładających się fragmentów graczy. Znaczenie poszczególnych bitów:

0 Gracze 0-3, pola gry 0-3, tło.
1 Gracze 0-1, pola gry 0-3, gracze 2-3, tło.
2 Pola gry 0-3, gracze 0-3, tło.
3 Pola gry 0-1, gracze 0-3, pola gry 2-3, tło.
4 Cztery pociski tworzą dodatkowego piątego gracza.
5 Nakładające się fragmenty graczy przyjmują trzeci kolor.
6,7 Tryby graficzne GTIA.

Może się zdarzyć, że przypadkowo ustawimy bity GPRIOR tak, że wybierzemy dwa sprzeczne priorytety wyświetlania. W takim wypadku w miejscach, w których pokrywają się gracze pojawi się kolor czarny. Ten sam efekt otrzymamy jeśli nie ustawimy bitu piątego.

Zapisanie jedynki do piątego bitu umożliwia stworzenie wielokolorowych graczy. W miejscach gdzie nakładają się obiekty 0 i 1 oraz 2 i 3 (tylko w tych kombinacjach) pojawi się trzeci kolor, utworzony z dwóch kolorów graczy.

Więcej informacji o grafice graczy i pocisków w odpowiednim (jednym z przyszłych) odcinku cyklu.

Na szczególną uwagę zasługują bity 6 oraz 7 opisywanego rejestru. Zezwalają one na włączenie dodatkowych trybów graficznych GTIA. Tryby te zostały wprowadzone jako rozszerzenie możliwości graficznych małego ATARI wraz z pojawieniem się układu GTIA, który zastąpił przestarzały CTIA.

Sam ANTIC, o którym pisałem sporo w poprzednim odcinku mapy pamięci nie wystarczy do utworzenia na monitorze kompletnego obrazu. Nie generuje on bowiem kolorów, które często bywają bardzo potrzebne. ANTIC przesyła informacje o kolorach do wspomnianego już GTIA, który dopiero tworzy kolorowy wizerunek pewnego obszaru pamięci komputera. Odpowiednią kombinacją bitów 6 i 7 komórki GPRIOR można wybrać sposób przedstawiania danych przez GTIA. A oto zestawienie możliwych rodzajów wyświetlania.

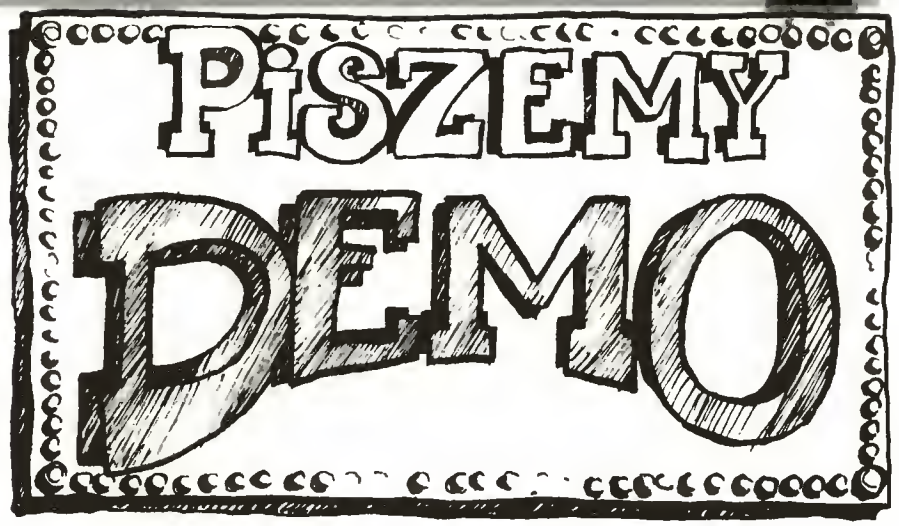
00 Normalne wyświetlanie. Ta kombinacja bitów umożliwia uruchomienie starych programów dla ATARI XL/XE ponieważ jest właściwa dla układu CTIA.
01 Taka kombinacja bitów odpowiada trybowi graficznemu 9 BASIC-u. Dane obrazu zbierane są w czterobitowe kawałki interpretowane jako stopnie jasności koloru z rejestru COLBAK. Np. w trybie wysokiej rozdzielczości GTIA wyświetla paski o tej samej wysokości, lecz czterokrotnie szersze od normalnych punktów.
10 Taka kombinacja bitów odpowiada trybowi graficznemu 10 BASIC-u. Czterobitowe kawałki, na jakie podzielona zostaje pamięć obrazu są interpretowane jako numery rejestrów koloru. W teorii więc można by korzystać z 16 kolorów, lecz niestety GTIA dysponuje jedynie dziewięcioma.
11 Taka kombinacja bitów odpowiada trybowi graficznemu 11 BASIC-u. Czterobitowe kawałki, pobrane z pamięci obrazu są teraz interpretowane jako kody kolorów, których jasność jest pobierana z rejestru COLBAK.

Przedstawiony poniżej maleńki programik pozwoli nam na obejrzenie opisanych przed chwilą trybów GTIA.

```
10 GRAPHICS 0:SCRMEM=PEEK(88)+256*PEEK
(89)
20 FOR I=0 TO 319:POKE SCRMEM+I,RND(0)
*255:NEXT I
30 GOSUB 100
40 POKE 623,128
50 GOSUB 100
60 GRAPHICS 0:END
100 IF PEEK(764)=255 THEN 100
110 POKE 764,255:RETURN
```

W wierszu 40 wystarczy zmienić wartość 128 na 64 lub 192, aby zobaczyć pozostałe dwa tryby. Grafikę 0 w wierszu 10 również można zmienić na dowolną inną.

Jarosław Syrylak



Zamiast wstępu

Poniższy artykuł rozpoczyna cykl pod tytułem "Piszemy Demo". Mamy nadzieję, że po skompletowaniu i przeanalizowaniu wszystkich odcinków tego cyklu, każdy z zainteresowanych czytelników będzie w stanie zaprogramować swoje pierwsze demo. Zapewne większość z Państwa zauważyła już, że ten artykuł nie ma nic wspólnego z opublikowaną wcześniej zapowiedzią serii.

Co już umiemy?

Wszystkie programy publikowane w kolejnych artykułach naszego cyklu napisane zostały w QA. Z tego względu zakładamy, że każdy pragnący poznać tajemnice Atari posiada legalną (poprawnie działającą) kopię tego programu i wnikliwie przestudiował rozdział instrukcji poświęcony programowaniu 6502. Oprócz znajomości mnemoników i trybów adresowania naszego procesora czytelnik powinien umieć posługiwać się jeszcze dwoma systemami numeracji: dwójkowym i szestnastkowym.

Co to jest demo?

Demo, jak sama nazwa wskazuje, jest programem prezentującym (z jak najlepszej strony) możliwości graficzne i muzyczne komputera. W przypadku naszego 8-bitowego Atari, zadanie to nie jest proste: konieczna jest umiejętność programowania procesora graficznego ANTIC oraz znajomość rejestrów GTIA. Obydwa te układy odpowiedzialne są za tworzenie obrazu. POKEY natomiast, między innymi, tworzy dźwięk. W tym i w następnym artykule zajmiemy się szczegółowo ANTIC'iem – czyli ekranem.

Display List

ANTIC, jak każdy procesor, potrafi wykonywać program. Poszczególne rozkazy programu (omówione poniżej) określają strukturę wyświetlanego obrazu. Wektor DLPTRS umieszczony w pamięci w komórkach $230 i $231 (560 i 561) wskazuje na program ANTIC'a. ANTIC potrafi zaadresować 64 kilobajty pamięci, podobnie jak CPU (Central Processor Unit - centralna jednostka sterująca, czyli 6502). Oznacza to, że program ANTIC'a możemy umieścić w dowolnie wybranym miejscu pamięci.

Rozkazy ANTIC-a

Rozkazy mikroprocesora ANTIC można podzielić na trzy grupy:

- tworzące puste linie;
- tworzące linie trybów graficznych lub tekstowych;
- rozkazy skoków.

Rozkazy tworzące puste linie mają cztery młodsze bity skasowane. Bity 4-6 zawierają liczbę tworzonych pustych linii, zmniejszoną o jeden. Oto pełna lista rozkazów tworzących puste linie:

```
jedna pusta linia     - $00
dwie puste linie      - $10
trzy puste linie      - $20
cztery puste linie    - $30
pięć pustych linii    - $40
sześć pustych linii   - $50
siedem pustych linii  - $60
osiem pustych linii   - $70
```

Następną wyróżnioną przez nas grupą rozkazów są rozkazy tworzące linie trybów graficznych lub tekstowych. Poniższa tabela zawiera zestawienie wszystkich tych trybów:

tryb ANTIC-a	tryb OS
$02	$00
$03	-
$04	$0C
$05	$0D
$06	$01
$07	$02
$08	$03
$09	$04
$0A	$05
$0B	$06
$0C	$0E
$0D	$07
$0E	$0F
$0F	$08

Wykorzystując to, co wiemy, ułóżmy pierwszy program dla ANTIC'a, wyświetlający na ekranie dziewięć linii w trybie $02 ANTIC'a:

```
Program Equ $8800

List_mem Equ %00000110
List_err Equ %00000101
Code_mem Equ %00010000
Code_dsk Equ %00100000

Dlptrs   Equ $0230
Kbcodes  Equ $02FC

         Opt List_err+Code_mem

         Org Program


         Lda Dlptrs+1
         Pha
         Lda Dlptrs
         Pha
         Ldx <DList
         Ldy >DList
         Stx Dlptrs
         Sty Dlptrs+1
         Jsr Klawisz
         Pla
         Sta Dlptrs
         Pla
         Sta Dlptrs+1
         Rts

Klawisz  Lda #$FF
         Sta Kbcodes
L1       Cmp Kbcodes
         Beq L1
         Rts

DList    Equ *
     Dta B($02),B($02),B($02)
     Dta B($02),B($02),B($02)
     Dta B($02),B($02),B($02)

         End
```

Po uruchomieniu programu wektor DLPTRS odkładany jest na stos. Następnie ustawiamy go tak, aby wskazywał na nasz program ANTIC'a, zaczynający się od etykiety DList. Podczas gdy komputer oczekuje na naciśnięcie klawisza (procedura Klawisz) widoczny jest efekt działania programu. Po naciśnięciu klawisza przywracany jest stary wektor DLPTRS. Zgodnie z oczekiwaniami na ekranie pojawiło się dziewięć linii trybu $02 ANTIC'a. Oprócz nich zostały wyświetlone również inne, zupełnie przypadkowe linie obrazu, ponieważ program ANTIC'a nie został poprawnie zakończony. Aby to zrealizować musimy poznać jeszcze dwa rozkazy, rozkazy skoków:

- JMP ($Ø1) - tworzy jedną pustą linię, a następne dwa bajty programu ANTIC'a nie są rozkazami, tylko adresem dalszej części Display List. Rozkaz ten jest analogiczny do rozkazu JMP procesora 6502.
- JVB ($41) - kończy program ANTIC'a, a kolejne dwa bajty programu nie są rozkazami, tylko powinny być adresem początku Display List.

Teraz jesteśmy w stanie zlikwidować przypadkowe linie, umieszczając na końcu programu ANTIC'a rozkaz JVB. Część programu zawierająca Display List ma postać:

```
Dlist   Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($41),A(Dlist)

        End
```

W ten sposób otrzymaliśmy dziewięć linii trybu $02 ANTIC'a. Linie te zawierają przypadkowe znaki. Dzieje się tak dlatego, że nie poinformowaliśmy ANTIC'a skąd ma pobierać dane do wyświetlania. Tak więc, musimy się zapoznać z kolejnym pojęciem LMS (Load Memory Scan). Zmodyfikujmy poprzedni program tak, aby ANTIC pobierał dane z wyznaczonego przez nas obszaru pamięci. Program po modyfikacji, od etykiety DList, wygląda następująco:

```
DList   Dta B($42),A(Text)
        Dta B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($41),A(Dlist)

        Org $AØØØ

Text    Dta D'Tajemnice Atari.'

        End
```

W pierwszym rozkazie programu ANTIC'a ustawiony został szósty bit (czyli do $02 dodaliśmy $40), a następujący po nim wektor wskazuje na wybrany przez nas obszar pamięci. Z obszaru tego ANTIC będzie pobierał dane wyświetlane na ekran przy pomocy licznika pamięci obrazu. Licznik ten jest automatycznie zwiększany przez ANTIC. Przy tej operacji brane jest pod uwagę tylko 12 najmłodszych bitów licznika, co ogranicza zakres jego działania do czterech kilobajtów. W praktyce oznacza to, że jeśli dane obrazu zajmują więcej niż 4 kilobajty, to po przekroczeniu granicy czterokilobajtowego bloku kolejne dane będą pobierane od początku tego bloku. W naszym przykładzie licznik pamięci ekranu ustawiony jest na tekst rozpoczynający się od etykiety Text. Po uruchomieniu programu tekst ten ukaże się na ekranie. Zwróćmy uwagę na to, że napis "Tajemnice Atari" zapisany jest w pamięci w kodzie IOCODE. Na koniec pierwszego artykułu podajemy jeszcze jedną proponowaną przez nas modyfikację. Ponieważ linie wyświetlane są od samej góry ekranu i mogą być niewidoczne na niektórych monitorach, to należałoby obniżyć wyświetlany ekran. W tym celu umieśćmy trzy rozkazy $70 (tworzące w sumie 24 puste linie). Każdy program ANTIC'a tworzony przez system operacyjny rozpoczyna się tymi rozkazami. Oto zmodyfikowany fragment naszego programu:

```
DList   Dta B($7Ø),B($7Ø),B($7Ø)
        Dta B($42),A(Text)
        Dta B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($Ø2),B($Ø2),B($Ø2)
        Dta B($41),A(Dlist)

        Org $AØØØ

Text    Dta D'Tajemnice Atari.'

        End
```

Dalszy ciąg za miesiąc...

Tomasz Bielak
Rafał Bielecki

Efektem działania tego "podglądacza", w odróżnieniu od innych tego typu programów, nie jest wyświetlanie samego tylko numeru linii. Tutaj użytkownik widzi, krok po kroku, które wiersze programu zostaną wykonane, i może zaobserwować efekt ich działania.

Program umieszczony jest od adresu $600. Przepisuje on ROM do RAM i podmienia procedurę PRCSTM w interpreterze BASIC-a. Wykorzystano tu procedurę Andrzeja Zalewskiego opublikowaną w czasopiśmie "Moje Atari" nr 3/91.

W czasie działania programu TRACE przed wykonaniem każdego wiersza (także w trybie bezpośrednim) komputer czeka na decyzję użytkownika. ESC powoduje niezwłoczne wykonanie tego wiersza. RETURN prowadzi do ekranu podglądowego, gdzie widać tekst wiersza przeznaczonego do wykonania. Następny RETURN spowoduje wykonanie tego wiersza, zaś DELETE - przerwanie programu.

Wojciech Rawdanik

Od redakcji: Podczas redakcyjnych testów program nie zatrzymywał się w pierwszym wierszu, bezpośrednio po RUN. Co na to autor?

```
HM  3 REM PROCEDURA TRACE DLA ATARI
YU  4 REM AUTOR:W.RAWDANIK
NK  5 REM
XT  6 REM ROM DO RAM  (A.ZALEWSKI)
NM  7 REM
ZO  10 CLR :FOR N=1536 TO 1593:READ A:POKE
     N,A:S=S+A:NEXT N:AD=40960:? CHR$(125)
JW  20 POKE 1556,253:X=USR(1537,AD,24576)
PV  30 POKE 1556,254:X=USR(1537,24576,AD)
OA  40 AD=AD+256:IF AD>=53248 AND AD<55000
     THEN AD=55296
IL  50 IF AD>=65535 THEN 70
RW  60 GOTO 20
YR  70 POKE 54017,254
BG  71 REM
YC  72 REM PROCEDURA TRACE
BK  73 REM
JG  75 RESTORE 200
FY  80 FOR T=1536 TO T+254:READ D:POKE T,D
    :NEXT T
BH  81 REM
NF  82 REM PROCEDURKA URUCHAMIAJACA
BL  83 REM
KF  85 RESTORE 400
GW  90 FOR T=32768 TO T+11:READ D:POKE T,D
    :NEXT T
ZK  94 ? CHR$(125):? :? :? "Procedura TRAC
    E zainstalowana ":? :? "ESC..... norma
    lnie":? "RETURN ..... wyswietlanie lin
    i"
BA  95 ? "DELETE ..... przerwanie programu
    "
ZK  96 ? "UWAGA ! ROM znajduje sie w pamie
    ci RAM"
BU  97 REM
BC  98 REM DANE PROCEDURY ROMRAM
BY  99 REM
MK  100 DATA 64,104,104,141,44,6,104,141,4
    3,6,104,141,47,6,104,141,46,6,120,169,
    254,141,1,211,169,0,141
ZD  110 DATA 250,255,141,254,255,169,6,141
    ,251,255,141,255,255,162,0,189,120,230
    ,157,64,156,232,208,247,88,169
FN  120 DATA 253,141,1,211,96
RF  125 REM
WS  130 REM DANE PROCEDURY TRACE
RH  135 REM
UK  200 DATA 160,255,140,252,2,173,252,2
ZG  202 DATA 201,12,240,7,201,28,208,245
IC  204 DATA 76,171,6,140,252,2,140,240,6
NG  206 DATA 173,48,2,72,173,49,2,72,173
MK  208 DATA 197,2,72,173,198,2,72,173
UO  210 DATA 244,2,72,173,111,2,72,41,63
GZ  212 DATA 141,111,2,169,241,141,48,2
NZ  214 DATA 169,6,141,49,2,169,202,141
AU  216 DATA 197,2,169,148,141,198,2,169
VQ  218 DATA 224,141,244,2,169,0,160,160
PY  220 DATA 153,255,206,136,208,250,169
CW  222 DATA 76,141,153,186,169,184,141
KG  224 DATA 154,186,169,6,141,155,186,32
CK  226 DATA 142,181,169,0,172,240,6,153
MF  228 DATA 0,207,169,166,141,153,186
IB  230 DATA 169,181,141,154,186,169,72
NU  232 DATA 141,155,186,160,255,173,252
JX  234 DATA 2,201,12,240,5,201,52,208
CB  236 DATA 245,200,104,141,111,2,104
VT  238 DATA 141,244,2,104,141,198,2,104
ER  240 DATA 141,197,2,104,141,49,2,104
GQ  242 DATA 141,48,2,152,240,6,140,252,2
SI  244 DATA 76,25,184,136,140,252,2,76
MP  246 DATA 146,183,162,0,142,239,6,32
BX  248 DATA 203,6,172,240,6,200,153,0
DN  250 DATA 207,140,240,6,96,16,8,162
XO  252 DATA 128,142,239,6,56,233,128,201
BN  254 DATA 31,176,7,24,105,64,13,239,6
ZY  256 DATA 96,201,95,176,7,56,233,32,13
RK  258 DATA 239,6,96,13,239,6,96,0,0,112
RN  260 DATA 66,0,207,2,2,2,65,241,6
RD  280 REM
OP  285 REM DANE PROCEDURY URUCHAMIAJACEJ
RF  290 REM
XW  400 DATA 104,169,0,141,95,169,169,6,14
    1,96,169,96
IJ  415 A=USR(32768)
```

VOO DOO CASTLE

Mało który z początkujących łamaczy joystick'a zwraca więcej niz 2 minuty uwagi na gry przygodowe. Dopiero, gdy gry zręcznościowe już nie bawią, amator gier odkrywa powoli świat gier strategicznych i przygodowych. Gry przygodowe zwane też "tekstowymi" polegają na sterowaniu przebiegiem akcji za pomocą słownych poleceń. Mogą one dostarczyć chyba najwięcej przyjemności i satysfakcji, gdy potrafimy je ukończyć. Nie są przy tym ani w połowie tak trudne, jak to uważają niewtajemniczeni. W większości gier korzysta się z bardzo ograniczonego słownictwa rzędu 100 – 300 słów. Polecenia wydawane przez gracza komputerowi są najczęściej żywcem wzięte ze słownika Kali'ego. Rozkazujemy: "brać przedmiot", "iść wschód" itd. Nie jest więc potrzebna znajomość gramatyki.

Jedną ze stosunkowo prostych słownikowo, a dających możliwość przeżycia wielkiej przygody bez opuszczania swojego ulubionego fotela jest właśnie "Voo Doo Castle". W grze można wykorzystać około 120 słów. Polecenia składają się z co najwyżej dwóch słów, z których pierwsze to najczęściej czasownik. Słowa można skracać do 3 liter. Niektóre często wykorzystywane zwroty da się skrócić jeszcze bardziej. Polecenia wyboru kierunku "idź na południe, północ" itp. można skrócić do pojedynczych liter oznaczających kierunki N, S, E, W (od North – północ, South – południe, East – wschód, West – zachód). Na każdym etapie gry możesz nagrać jej stan na dyskietkę poleceniem Save Game, a także zrezygnować z gry (Quit). Można sprawdzić jakie przedmioty aktualnie niesiesz (Inventory). Każdy obiekt da się wziąć (take), wyrzucić (drop), zbadać (examine ...) lub obejrzeć (look ...). Każda plansza informuje nas jakie są w niej przedmioty (Visible items) oraz wskazuje możliwości opuszczenia tej komnaty (Some exits are NS...– niektóre wyjścia są na północ, południe ...).

Stop! Może nie psuj sobie zabawy? Odłóż teraz TA, uruchom grę i spróbuj samodzielnie coś zdziałać. Dopiero gdy Ci braknie pomysłów, sięgnij do poniższego tekstu...

Grę rozpoczynasz w kaplicy (a Chapel). Widzisz zamkniętą trumnę (Closed Coffin). Akcja nasuwa się sama - otwierasz ją (Open Coffin). Na trumnie znajduje się napis "Count Cristo's been cursed! There's one way for him to flee! Find it, and he'll go free!" (Książę Cristo został zaklęty w kamień, lecz jest sposób, by go uwolnić! Znajdź go!). Widzimy także mężczyznę mającego na ręce pierścień z szafirem (a man wearing a sapphire ring). Spróbujemy zabrać mu go (Take Ring). Udało się. Sprawdzimy pierścień (Examine Ring). Okazuje się, że na pierścieniu jest coś napisane (There's something written here). Musimy to przeczytać (Read Ring). Napis brzmi: "pomachaj mną" (Wave Me). Próbujemy (Wave Ring) i nic. Może jeszcze obejrzymy mężczyznę (Look man) i zaczniemy zwiedzanie zamku.

Najpierw spróbujmy pójść na północ (N). Znajdujemy się w jednej z komnat zamku, kruki krzyczą coś na zewnątrz (Raven crys something outside), a okno w komnacie właśnie się zatrzasnęło. Spróbujemy je otworzyć (Open Window), rozbić (Break Window), niestety... Posłuchamy zatem, co mają do powiedzenia ptaszyska (Hear Raven). Mówią one, by zapytać u sprzedawcy o grę "książę" (Count). Będzie to miłość od pierwszego bajtu (it will be love at first byte). Ale reklama nic nie wnosi do gry. Skoro nic nie zdziałaliśmy, wracamy do kaplicy (S) i udajemy się na wschód (E).

Znajdujemy się w tunelu. Widzimy skrwawiony nóż (Bloody Knife) oraz masywne drzwi z ogromnym szafirem (massive stone door with a sapphire set into it). Zabieramy nóż (Take knife). Drzwi nie chcą się otworzyć więc może spróbujemy podłubać coś kozikiem (Cut Door). Niestety nóż nie robi na nich żadnego wrażenia (Knife made no impression on it). Ale...ten szafir kojarzy się nam z pierścieniem. Może pomachamy nim jeszcze raz. Tak, to jest to - pierścień rozbłysł i drzwi znikają (Ring glows briefly & door vanishes).

Widzimy teraz jamę prowadzącą w dół (slick chute leading downward). Pójdziemy tam (Go chute). W jednej z komnat zamku znajdujemy plakietkę, (Plaque), jamę prowadzącą w górę, pęk-

nięcie (Crack) i dziurę w ścianie (hole in the wall). Na początek sprawdzamy plakietkę (Examine Plaque). Okazuje się, że są tam napisy. Próby czytania (Read Plaque) są nieudane. Litery są zbyt małe, by coś dojrzeć nieuzbrojonym okiem. No cóż, weźmiemy ją ze sobą (Take Plaque). Spróbujemy wejść w pęknięcie (Go Crack). Jesteśmy jednak zbyt duzi, by to zrobić bez użycia potężnej magii. Wchodzimy zatem do dziury w ścianie (Go Hole).

Znajdujemy się w pokoju wróżki Meagen, lecz ją samą wystraszyliśmy i schowała się (You startled medium & she vanished) Widzimy kryształową kulę (crystall ball). Rozbić ją chyba szkoda (Break ball) lecz moze sobie w nią popatrzymy (Look ball). Lecz co to? Zostałem przeniesiony z powrotem do tunelu (Spirit vibration drive me from room). Widocznie wróżka nie lubi, gdy ktoś buszuje w jej pokoju.

Wracamy do kaplicy (W) i udajemy się na południe (S). Znajdujemy się w korytarzu (Dingy looking Stairwell). Widać pęknięte szkiełko (Broken glass) i schody. Weźmiemy szkiełko (take glass) i spróbujemy przeczytać co jest napisane na plakietce. Tym razem jednak światło jest zbyt jaskrawe, by przeczytać fosforyzujące litery (Phosphorescent letters are very hard to read in bright light). Poszukamy szczęścia na zachodzie (W). Znajdujemy małą łopatę (Pocket shovel). Widać także otwarte drzwi lochów (jail cell). Ani nam się śni zgnić w lochach, idziemy zatem na południe. Znajdujemy się w sali tortur (torture chamber). Widać otwarte, malutkie drzwi. No cóż, nie jesteśmy krasnoludkami. Idziemy na wschód do zbrojowni (Armory). Zabieramy stąd tarczę (shield) i złamany miecz (broken sword). Nie mając pomysłów co dalej robić, wracamy na korytarz.

Udajemy się na wschód. Jesteśmy w jednym z pokojów zamku. Widoczny duży kocioł (big kettle) możemy zbadać, co pozwala ujrzeć znajdującą się pod nim dziurę. W kotle jest trochę zupy (soup). Zjemy ją (eat soup). Od razu poczuliśmy się lepiej. Próbujemy pchać, ciągnąć ten wredny kocioł – i nic. Zdenerwowani posyłamy mu kopa (kick kettle). Pomogło – garnek się przesuwa. Wchodzimy do dziury. Jeat tu bardzo ciemno, możemy zatem przeczytać napis na plakietce (read plaque). Napis brzmi "Safe – 38 33" (sejf 38 33). Na nic teraz ta informacja – wychodzimy zatem (Up). Otrzymujemy też ostrzeżenie, że poruszanie się w ciemności jest niebezpieczne.

Idziemy na północ. W tym pokoju na ścianach znajdują się trofea (animal heads). Weźmiemy je i widzimy ukryty sejf (Safe). Sprawdzamy go. Jest on zamknięty kombinacją cyfr od 33 do 38 (Safe's combination lock is from 33 to 38). Pamiętamy co głosił napis na plakietce. Kręcimy więc 38 (Turn 38) i 33 (Turn 33). Sejf jest otwarty Zabieramy znajdujący się w nim stary młotek (antique hammer).

Teraz na wschód. Znaleźliśmy żelazny czajnik do odprawiania czarów (Cast iron pot). Sprawdzenie go wykaże, że znajduje się tam wywar czarownic. Dalej na wschód i jesteśmy w laboratorium. Widzimy torbę czarodziejską pana Ju–Ju, probówki (test tubes) i etykietowane chemikalia (labelled chemicals). Próbujemy wziąć torbę – mamy jednak wrażenie jakby była przyśrubowana do podłogi. Przy okazji jedna z probówek eksploduje (one of the test tubes exploded) Dobrze, że mamy tarczę. Okazuje się, że niesiemy za dużo przedmiotów przy sobie Musimy trochę wyrzucić. Wyrzucamy więc szkiełko (drop glass), plakietkę, pierścień – czyli niepotrzebne juz przedmioty. Zabieramy odczynniki i probówki. Sprawdźmy to wszystko.

Chemikalia wyglądają tak, jak gdyby miały zostać zmieszane ze sobą. Mieszamy je zatem (Mix chem). Teraz wyglądają prawie jak lemoniada (looks almost like lemonade). No to cóż, pijemy? Pijemy (Drink chem). Grzmot, i okazuje się, że zmaleliśmy na 4 stopy. To dobrze – minęliśmy kilka miejsc, gdzie trudno było się zmieścić. Wracamy na korytarz i zabieramy stamtąd łopatę.

Idziemy dalej do sali tortur. Drzwi są teraz szeroko otwarte (wide open door). Wchodzimy w nie (Go door) i jesteśmy na cmentarzu (graveyard), widzimy groby (graves) oraz zardzewiałą piłę (rusting saw). Sprawdzamy dokładnie groby. Znajdujemy czterolistną koniczynę (four leaf clover). Okazuje się również, że jest tu grób zarezerwowany dla nas (this grave reserved for You) Bierzemy koniczynkę i piłę. Oczywiście musimy wyrzucić jakiś przedmiot. Wyrzućmy miecz. Mamy łopatę, więc moze coś wykopiemy? Próbujemy (dig grave). Słyszymy głos "strange hobby You have" (masz dziwne hobby). Po powrocie do pokoju z oknem okazuje się, że jest ono otwarte. Zostawiamy tu koniczynkę i wchodzimy na okno (go window).

Na parapecie znajdujemy lalkę (Doll). Okazuje się, że lalka przedstawia księcia Cristo i jest poprzebijana szpilkami (Doll looks like count Cristo. There're pins in it). Nie da się ich wyjąć (pull pins) w tej chwili (I cant - yet). Wracamy do kaplicy i udajemy się na zachód do bawialni. Widać duży kominek (large fireplace). Wchodzimy doń (go fireplace). Znajdując się w nim widzimy zamknięcie komina (Close flue) oraz zakurzoną statuetkę przedstawiającą bożka (dusty idol). Skoro brudny, więc oczyśćmy go (Clean idol) Bożek zaczyna świecić (idol begins to glow). Otwieramy wejście do kominka (open flue), zabieramy ze sobą bozka i wchodzimy tam (go flue). Możemy tam znaleźć sadzę (soot) i widzimy deski przybite do ścian komina (Wooden boards nails to chimney). Wyjmujemy gwoździe (take nails) i zabieramy deski (take boards). Pod deskami znajdowała się krata (grating). Przecinamy ją za pomocą piły (saw grating). To ciężka praca, lecz w końcu dajemy sobie radę. Pod kratą znajduje się przycisk. Naciskamy go (push button) i silny podmuch wyrzuca nas do góry (duty exhaust fan comes on & sucks me up).

W wąskiej części komina znajduje się uwięziony kominiarz (Chimney sweep) Pomagamy mu się wydostać (push sweep). Z wdzięczności za uwolnienie wciska nam przed zniknięciem kawałek papieru (hands me a piece of paper) Czytamy w nim: "Powiedz Zap, by przywrócić kogoś zakiętego w kamień" Wychodzimy z komina i kierujemy się w stronę korytarza. Tutaj wchodzimy na schody (go stairs). Znajdujemy się w gabinecie (a parlor). Widać tu rzeźbę (Ju–Ju statue). Zabieramy ją i idziemy do kaplicy. W kaplicy wypowiadamy słowa zakięcia (say zap). Grzmoty i rzeźba zaczyna pękać. Mamy wrazenie, że ktoś jest z nami w tym pomieszczeniu – słyszymy czyjeś jęczenie (I hear someone mumbling). Słuchamy więc (hear man). Ju–Ju man mówi "Moja torba jest teraz twoja, jej magia może Ci pomóc" (My bag is now Yours. Its magic will help You – Crack the course!). Idziemy do laboratorium, otwieramy torbę i znajdujemy tam książkę (a book) oraz czarodziejaką różdżkę (stick). Książka zawiera słowa "With knife in hand... Circle Coffin and..." (z nożem w ręku okrąż trumnę i ... – reszta strony jest urwana.

Wracamy do kaplicy zabierając jeszcze z komnaty, znajdującej się pod kotłem, symbol pomyślności – zajęczą stopkę (rabbit's foot). Rzucamy zajęczą stopkę (drop foot) na księcia (on man). Słychać grzmot. Okrążamy trumnę (Circle Coffin) – następny grzmot. Zrobiło się ciemno. Jedyne światło pochodzi od świecącego bożka (Its very dark, the only light is from the idol)...

Co dalej? Tego nie powiem. Pierwsza osoba, która odnajdzie sekretne przejście Ju–Ju man'a, zobaczy uśmiechniętego księcia i uzyska od niego skarby, krótko mówiąc: ukończy grę, może po opisaniu dalszego ciągu otrzymać nagrodę. Będzie to wybrana praz zwycięzcę gra sprzedawana przez firmę Avalon.

Jaro

FORTH

część 4

Na początku tego odcinka chciałbym zacytować fragment listu, jaki otrzymałem od pewnego właściciela Amigi. Była to odpowiedź na moje pytanie o język FORTH na ten komputer.

Zacytuję: *Chciałem Pana poinformować, że języki FORTH, FORTRAN nie istnieją w wersji na Amigę. Myślę że nie ma sensu programować w języku FORTH, jeżeli on praktycznie wychodzi z użytku.*

Mam nadzieję, że użytkownicy komputerów ATARI XL/XE nie podzielają poglądów zawartych w tych słowach. Aby rozwiać całkowicie wszelkie wątpliwości, postanowiłem skomentować powyższy fragment listu.

Po pierwsze:

Języki FORTH i FORTRAN łączą cztery rzeczy: F, O, R i T. Poza tym, są to języki całkowicie odmienne.

Po drugie:

Sądzę, że język FORTH nie zdążył jeszcze dobrze wejść do użytku, aby miał już z niego wychodzić. Ostatni standard pochodzi z roku 1983 i trudno tu mówić o czymś przestarzałym. A swoją drogą: oczywiście, język FORTH istnieje w wersji na Amigę.

Przystąpmy jednak do kolejnej odsłony, ukazującej tajniki naszego języka.

ZADANIE

Przed poznaniem kolejnych słów postawmy przed sobą następujące zadanie: Napisać słowo o nazwie PAUSE, zatrzymujące program na ustalony czas. Parametrem wejściowym tego słowa będzie liczba określająca czas postoju. Większość czytelników domyśla się zapewne, że słowo to stworzymy w oparciu o wewnętrzny zegar naszego komputera.

PĘTLE

Aby wykonać postawione zadanie, musimy poznać podstawową strukturę każdego języka programowania. Jest nią pętla.

Zacznijmy od tej najprostszej, a przy tym – najmniej pożytecznej. Ma ona następującą postać:

```
BEGIN ... AGAIN
```

Pętla ta wykonuje bez końca wszystkie słowa zawarte między BEGIN a AGAIN. Uwaga! Wszystkie pętle w języku FORTH muszą występować wewnątrz zadeklarowanego słowa. Niedopuszczalne jest użycie takiej pętli w trybie bezpośrednim, np:

```
BEGIN 1 . AGAIN RETURN
```

Każda pętla musi mieć swój początek i koniec wewnątrz jednego słowa, np:

```
: PET BEGIN 1 . AGAIN ; RETURN
```

Wykonajmy teraz nasze słowo pisząc:

```
PET RETURN
```

Słowo to będzie się wykonywało w nieskończoność i nie pomoże nam naciskanie klawisza BREAK. Jedynym ratunkiem jest niezastąpiony RESET, po którym FORTH wróci do stanu początkowego.

Następną pętlą, którą chciałbym przedstawić jest tzw. pętla liczona. Ma ona następującą postać:

n2 n1 DO ... LOOP

n1 – to liczba określająca wartość początkową licznika pętli, znajdująca się na szczycie stosu

n2 – wartość końcowa licznika pętli, znajdująca się na stosie pod *n1*

DO – otwiera pętle liczoną

... – tu znajdują się dowolne słowa wewnątrz pętli

LOOP – zamyka pętlę liczoną

Oczywiście, pętla ta może występować tylko wewnątrz zadeklarowanego słowa. Przykład:

```
: PEL 20 0 DO 1 . LOOP ; RETURN
```

i sprawdzamy:

```
PEL RETURN
```

Na ekranie ukazało się 20 jedynek. Aby uzyskać dostęp do licznika pętli wprowadzono słowo I. Wywołanie tego słowa (tylko wewnątrz pętli liczonej) zostawia na stosie

aktualny stan licznika pętli. Przykład:

```
: PEI 20 0 DO I . LOOP ; RETURN
```

i wywołujemy:

```
PEI RETURN
```

Na ekranie ukazały się liczby od 0 do 19, a nie do 20, jak byśmy się spodziewali.

Pętla była wykonywana, dopóki licznik nie osiągnął wartości końcowej, czyli w naszym przypadku 20. Wartość licznika jest zwiększana w słowie LOOP, tam też jest sprawdzany warunek końcowy. W przypadku osiągnięcia przez licznik wartości końcowej pętla zakończy się.

FORTH rozporządza także bardziej skomplikowaną odmianą pętli liczonej. Ma ona następującą postać:

n2 n1 DO ... *n* +LOOP

n2 i *n1* –to liczby na stosie mające identyczne znaczenie jak w poprzedniej pętli,

DO – otwiera pętlę

... – słowa wewnątrz pętli

n – liczba określająca skok licznika pętli. Musi zawsze występować przed **+LOOP** a może przyjmować dowolne wartości, także ujemne.

+LOOP – słowo kończące pętlę

Podczas korzystania z tej pętli należy uważać, aby licznik zbliżał się do górnej granicy, a nie oddalał.

Błędny byłby zapis:

```
: PEM 20 0 DO I . –1 +LOOP ;
```

Wartość początkowa licznika wynosi 0, a z każdym krokiem pętli licznik jest zwiększany o –1, czyli zmniejszany. Licznik zamiast zbliżać się do górnej granicy, oddala się od niej. Pętla ta nie zostanie nigdy zakończona.

Poprawny będzie jednak następujący zapis:

```
: PEM 0 20 DO I . –1 +LOOP ; RETURN
```

Wartością początkową jest 20, a końcową 0. Słowo to po wywołaniu wyświetli liczby od 20 do 1.

Skoro poznaliśmy już pętle liczone, przeprowadźmy test sprawdzający szybkość języka FORTH na naszym komputerze.

Deklarujemy słowo TEST:

```
: TEST 30000 0 DO LOOP ; RETURN
```

Wywołując słowo ze stoperem w ręku, możemy określić czas jego wykonania. Gdy ukaże się napis OK, oznacza to zakończenie działania słowa.

Aby uzyskać większą dokładność pomiaru czasu, możemy wykorzystać stworzone w poprzednich odcinkach słowa: 0ZEGAR! i ZEGAR@ .

Piszemy sekwencję:

```
0ZEGAR! TEST ZEGAR@ . RETURN
```

Na ekranie pojawiła się liczba, która podzielona przez 50 da nam czas w sekundach.

Tak, to nie przywidzenie, pętla wykonywała się około 3 sekund. Dla porównania, podobna pętla (FOR NEXT) w języku ATARI BASIC jest liczona 58 sekund.

PORÓWNANIA

Aby przedstawić inne rodzaje pętli, muszę zapoznać czytelników ze słowami wykonującymi porównania. Zacznijmy od słowa =

Operator ten pobiera i porównuje dwie liczby będące najbliżej szczytu stosu. Na ich miejscu zostawia jedną liczbę będącą tzw. znacznikiem porównania. Znacznik ten przybiera wartość 0 w przypadku fałszu i 1 dla prawdy.

W języku FORTH każda liczba niezerowa jest znacznikiem prawdy. Należy o tym pamiętać, gdyż jak przekonamy się dalej, jako znacznik można wykorzystać wynik z dowolnego działania matematycznego.

Kolejnymi słowami porównania będą operatory:

< – mniejszości
> – większości

Słowa te działają na stosie bardzo podobnie do operatora równości, lecz zostawiany wskaźnik jest obliczany następująco:

Dla < znacznik przybiera wartość 1 (czyli prawdy), gdy liczba znajdująca się niżej na stosie jest mniejsza od liczby będącej na jej szczycie.

Dla > znacznik wyniesie 1 (prawda), gdy liczba w głębi stosu jest większa od liczby na szczycie.

Sprawdźmy to w praktyce pisząc na komputerze:

```
10 5 = . RETURN
```

Otrzymamy wynik 0. Jeszcze raz:

```
10 10 = . RETURN
```

Tym razem otrzymamy oczywiście 1.

Dla nierówności:

```
10 5 > . RETURN
```

jest 1, a dla

```
10 5 < . RETURN
```

oczywiście 0.

Zauważmy, że znaki porównania wystąpiły w zapisie za liczbami, a nie między nimi. Wynika to z zastosowanej notacji zapisu działania (tzw. Odwrotnej Notacji Polskiej).

Poznane dotychczas porównania interpretują dane wejściowe jako liczby ze znakiem. Aby umożliwić porównanie liczb bez znaku, należy zastosować słowo U< . Działa ono podobne jak <, lecz nie uwzględnia znaku liczby. Pamiętajmy o tym, że każda liczba większa od 32767 jest interpretowana przez większość słów jako ujemna. Była o tym mowa w poprzednich odcinkach tego cyklu. Spróbujmy:

```
20000 45000 < . RETURN
```

Otrzymaliśmy wynik 0, a powinno być 1. Tak, lecz słowo < interpretuje liczby jako ujemne o całkowicie innych wartościach.

Sprawdźmy, jak widzi te liczby słowo <. Piszemy:

```
20000 . RETURN
```

i

```
45000 . RETURN
```

Widzimy, że nasze słowo zachowało się całkowicie poprawnie.

Ten sam warunek powinien wyglądać następująco:

```
20000 45000 U< . RETURN
```

Teraz otrzymaliśmy wartość 1.

Słowo U< znajduje szerokie zastosowanie przy porównywaniu adresów, które – jak wiemy – są liczbami dodatnimi.

Operatory przedstawione poniżej wymagają tylko jednego parametru wejściowego. Są nimi:

```
0= i 0<
```

Wykonują one porównanie liczby będącej na szczycie stosu z zerem.

Operator 0= pobiera liczbę ze szczytu stosu i zostawia na jej miejsce znacznik równy 1 (prawda), gdy liczba równa się 0, a 0 (fałsz), gdy jest inaczej.

Operator 0< zostawia na stosie znacznik prawdy (1) w przypadku, gdy liczba na stosie była ujemna.

Słowo 0= pełni w języku FORTH także funkcję operatora NOT, znanego nam z innych języków.

Umożliwia ono zamianę znacznika (prawda) 1 na (fałsz) 0 lub odwrotnie.

Czytelnicy zauważyli zapewne, że nie ma wśród standardowych słów języka FORTH słowa U> lub 0>. Co zrobić w przypadku, gdy zaistnieje konieczność wykonania takiego porównania? Nie ma z tym żadnych kłopotów, gdyż FORTH jest językiem umożliwiającym samodzielne tworzenie brakujących słów. Przykład:

```
: 0> 1 – 0< 0= ;
```

Zapewne zrozumiałe jest dla czytelników zastosowanie w prezentowanej deklaracji słowa 0<, a następnie zaprzeczenie znacznika przez 0=. Po co jednak odjęliśmy 1 od liczby na stosie? Gdyby opuścić tę operację, powstałoby następujące słowo:

```
: 0>= 0< 0= ;
```

Słowo to zostawiałoby znacznik prawdy także w przypadku, gdy liczba ze szczytu stosu równałaby się 0.

Mam nadzieję, że ze słowem U> czytelnicy poradzą sobie sami.

Kończąc opisywanie porównań, chciałbym zwrócić uwagę, że porównania wymagają zawsze parametrów na stosie. Niedopuszczalny byłby zapis:

```
10 5 < =
```

gdyż dla słowa = zabraknie jednego parametru. W takim przypadku należy zastosować:

```
10 5 > 0=
```

Aby zwiększyć czytelność możemy stworzyć odpowiednie słowo:

```
: <= > 0= ; RETURN
```

i zapisać:

```
10 5 <=
```

PĘTLE, CIĄG DALSZY...

Skoro poznaliśmy już operatory porównań, powróćmy do pętli. Kolejna z nich będzie miała postać:

```
BEGIN ... f UNTIL
```

BEGIN – początek pętli

... – jak w poprzednich przykładach

f – znacznik logiczny zostawiany na stosie

UNTIL – koniec pętli

Pamiętajmy, że znacznik *f* musi być zostawiany na stosie po każdym przebiegu pętli, gdyż słowo UNTIL pobiera go ze stosu.

Pętla ta jest wykonywana tak długo, aż znacznik *f* przybierze wartość różną od zera (prawda). Słowa będące wewnątrz pętli muszą ten znacznik odpowiednio modyfikować. Przykład:

```
: ES 255 764 C! BEGIN 764
C@ 28 = UNTIL ; RETURN
```

Zadeklarowaliśmy słowo o nazwie ES. Co ono wykonuje? Sprawdźmy:

```
ES RETURN
```

Nic się nie dzieje, dopiero naciśnięcie klawisza ESC powoduje opuszczenie słowa. Jedyną jego funkcją jest właśnie czekanie na naciśnięcie tego klawisza.

Ostatnia z prezentowanych pętli ma postać:

```
BEGIN ... f WHILE ... REPEAT
```

BEGIN – początek pętli

... – ciąg słów

f – znacznik logiczny na stosie

WHILE – słowo pobierające i sprawdzające znacznik. Gdy jest on równy 0 (fałsz), pętla jest opuszczana (skok za słowo **REPEAT**)

REPEAT – kończy pętlę

Zauważmy, że pętla ta różni się od poprzedniej, nie tylko odmiennym interpretowaniem znacznika, ale także dowolnością umieszczenia słowa WHILE sprawdzającego warunek. Jeżeli umieścimy WHILE na początku pętli (zaraz po słowie BEGIN), może ona nie być wykonana ani raz, podczas gdy BEGIN UNTIL musi się wykonać co najmniej raz. Przykład:

```
: AA 255 764 C! BEGIN 1 . 764 C@
255 = WHILE 2 . REPEAT ; RETURN
```

Nowo stworzone słowo **AA** wyświetla kolejno pary liczb 1 i 2. Gdy naciśniemy dowolny klawisz (komórka 764 przyjmie wartość różną od 255), pętla zostanie opuszczona. W ostatnim przebiegu pętli, słowa znajdujące się po WHILE nie będą wykonane (liczba 2 nie zostanie wyświetlona).

Sprostowanie

Do poprzedniego odcinka opowieści o języku FORTH wkradło się kilka nieścisłości.

Rada dla posiadaczy magnetofonów (str. 24) okazała się nietrafna. Zapisanie FORTH-a ze zmienioną wartością WARNING nic nie da, ponieważ FORTH odnawia jej stan po wprowadzeniu go do komputera. Należy więc każdorazowo ustawiać ją ręcznie.

Pod hasłem **Działania** pierwszy wiersz powinien wyglądać:

```
+ - * /      n1 n2 --- wynik
```

Złośliwy chochlik "złamał" ten wiersz na dwie części (inny złośliwy chochlik dopisał literkę i, która oczywiście nie jest symbolem działania.

Przepraszamy. Poszukiwania winnych trwają.

Redaktor

JESZCZE O STOSIE

Prezentowane w tej części artykułu słowa dotyczą stosu i nie mają w związku z tym odpowiedników w innych językach. Są to:

DUP

Słowo to tworzy duplikat liczby znajdującej się na szczycie stosu i zostawia go nad tą liczbą, powiększając stos.

SWAP

Zamienia między sobą dwie liczby znajdujące się najbliżej szczytu stosu.

DROP

Usuwa liczbę ze stosu.

OVER

Tworzy kopię liczby znajdującej się pod liczbą ze szczytu i umieszcza ją nad liczbą ze szczytu, powiększając odpowiednio stos. Liczby znajdujące się dotychczas na stosie nie ulegają zmianie.

ROT

Ustawia trzy liczby znajdujące się najbliżej szczytu w ten sposób, że trzecią wyciąga na szczyt, a dwie pozostałe obniża o jedno miejsce.

Przetestujmy działanie dwóch ostatnich słów, gdyż wydają się one najtrudniejsze do zrozumienia. Piszemy:

```
2 3 OVER . . . RETURN
```

Uzyskaliśmy ciąg liczb: 2 3 2. Liczba 2 została skopiowana nad dotychczasowym szczytem stosu, czyli liczbą 3. Następny przykład:

```
1 2 3 ROT . . . RETURN
```

Tym razem zobaczymy 1 3 2.

Pamiętajmy, że szczyt stosu przy wprowadzaniu znajduje się po prawej stronie ciągu i jest nim liczba 3. Przy wyświetlaniu – odwrotnie, gdyż najpierw będzie pobrana liczba ze szczytu. Widzimy więc, że liczba 1 przeskoczyła z trzeciego miejsca na szczyt stosu.

Jeżeli twój FORTH posiada słowo **S.** lub **.S** to możesz je stosować do przeglądania zawartości stosu.

REALIZACJA ZADANIA

Skoro poznaliśmy już wszystkie niezbędne struktury języka FORTH, możemy przystąpić do rozwiązania problemu postawionego na początku artykułu.

Do napisania słowa PAUSE wykorzystamy wcześniej stworzone słowa 0ZEGAR! i ZEGAR@. Były one opisane w poprzednich odcinkach tego cyklu. Więc do dzieła!

```
: PAUSE 0ZEGAR! BEGIN DUP
ZEGAR@ = UNTIL DROP ;
```

0ZEGAR! – zeruje stan zegara

BEGIN – otwiera pętlę

DUP – wykonuje duplikat parametru wejściowego

ZEGAR@ – zostawia na stosie stan zegara

= – porównuje wartość pobraną z komórek zegara z parameirem wejściowym

UNTIL – powoduje opuszczenie pętli, gdy znacznik logiczny pozostawiony na stosie przez = ma wartość niezerową (prawda)

Nasze nowe słowo PAUSE wymaga zawsze parametru wejściowego, np.:

```
50 PAUSE RETURN
```

Liczba występująca przed **PAUSE** określa czas działania słowa. Dla wartości równej 50 będzie on wynosił dokładnie 1 sekundę.

Sposób realizacji słowa **PAUSE**, przedstawiony przeze mnie, nie jest najlepszy, gdyż ingeruje w stan zegara.

Mam nadzieję, że czytelnicy potrafią sami rozwiązać ten problem w taki sposób, aby wartość komórek zegara nie ulegała zmianie.

Proszę więc o wasze rozwiązania słowa **PAUSE**, oczywiście w języku FORTH. Najlepsze z nich zostaną zaprezentowane na łamach TA.

Roland Pantoła

TOS

Prezentowany system turbo, w przeciwieństwie do innych tego typu programów dokładnie naśladuje pracę DOS–u tak, że prawie wszystkie programy pracują bez zarzutu. System jest głównie przeznaczony dla tych, którzy troszeczkę poważniej traktują małe ATARI.

Po przepisaniu dołączonego wydruku należy doczytać Zgrywusa z TA 1/92 (instrukcją ENTER), a po uruchomieniu wybrać opcję "format plikowy", a jako nazwę - C:. Można też użyć Mini-Zgrywusa, przedstawionego w TA 3/92. Po zgraniu programu na taśmę jesteśmy posiadaczami systemu turbo. Do szczęścia potrzebna jest jeszcze drobna przeróbka magnetofonu. Posiadaczy systemu TURBO 2000 ucieszy fakt że TOS jest z nim zgodny sprzętowo. Natomiast dla tych, którzy nie mają przerobionego magnetofonu, polecam układ opisany w czwartym numerze miesięcznika "MOJE ATARI".

TOS wczytujemy poprzez włączenie komputera (lub wykonanie zimnego startu) z wciśniętymi klawiszami START i OPTION lub z wciśniętym klawiszem START. W pierwszym przypadku po odczytaniu wszystkich rekordów komputer przez chwilę "pomruczy" i na ekranie pojawi się tekst zachęty T:. Natomiast w drugim pojawi się znajomy napis – READY. Aby wtedy uruchomić TOS, piszemy polecenie DOS i naciskamy RETURN. Wówczas pojawi się napis T:.

TOS jest gotowy do przyjmowania poleceń. Piszemy na przykład HLP i naciskamy RETURN. Na ekranie zostanie wyświetlona pełna nazwa programu i spis aktualnie dostępnych instrukcji. Do systemu można dołączyć jeszcze dwa polecenia, ale o tym w dalszej części opisu. Składnia każdej instrukcji jest następująca:

[u:]xxx [yyyy]

gdzie *u* – jednoliterowa nazwa urządzenia z obowiązującym po niej dwukropkiem, *xxx* – trzyliterowa nazwa polecenia, natomiast oddzielone spacją *yyyy* – ewentualną opcją. Fragmenty ujęte w nawiasy klamrowe można pominąć.

Na przykład: napiszmy **C:CAR ON**. Najpierw zostanie zmieniona bieżąca nazwa handlera na C: czyli na magnetofon w systemie NORMAL, a następnie zostanie wykonane podane polecenie. Możemy również napisać **CAR ON** – wykonane zostanie polecenie, ale bez zmiany handlera, lub CAR – przebieg polecenia będzie inny.

RUN

Napisana bez podania adresu (czyli bez dodatkowej opcji) uruchamia ostanio wczytany program. Jeżeli jest to niewykonalne, to następuje powrót do systemu. Napisana z adresem, który jest w notacji szesnastkowej powoduje, jak się nietrudno domyślić, start programu od podanego adresu. Na przykład: RUN E477 – zostanie wykonany zimny start komputera. Jeżeli naciśniemy RETURN wraz z klawiszem SHIFT – wykona się skok pośredni. Np. piszemy RUN 2E0 i naciskamy SHIFT+RETURN. Zostanie wykonany program, którego adres znajduje się w słowie S2E0. Jest to na ogół adres uruchomienia ostatnio wczytanego programu (niszczony po naciśnięciu klawisza RESET).

MOT

Powoduje włączenie lub wyłączenie silnika magnetofonu.

MEM

Wyświetla dolną i górną granicę wolnej pamięci RAM. Pojawia się napis np. Low = 15A4 High = BC1F – zależnie od tego czy są wczytane nakładki.

HLP

Poznana już wcześniej instrukcja, wyświetla nazwę programu oraz spis aktualnie dostępnych instrukcji.

CLS

Tą instrukcją możemy wyczyścić ekran.

Polecenia MOT, MEM, HLP, CLS nie mają dodatkowych opcji. Jeżeli po tych instrukcjach coś napiszemy (byle oddzielić spacją) to zostanie to odrzucone, a komenda wykona się normalnie.

DIR

Podobnie jak w DOS–ie, powoduje wyświetlenie nazwy programu. Z zainstalowanych handlerów jedynie sterownik T:, czyli turbo, ma możliwość zapisywania nazw. Jeżeli aktualne jest urządzenie T:, to po napisaniu DIR zostanie podany trzykrotny sygnał trąbki, wkładamy taśmę i naciskamy klawisz PLAY w magnetofonie (posiadacze przystawki TURBO 2000 powinni przełączyć magnetofon na system turbo), a następnie dowolny klawisz w komputerze. Po na trafieniu na początek pliku TOS wyświetli nazwę i znowu usłyszymy znajomy dźwięk. Polecenie się powtarza. Aby to przerwać, należy nacisnąć BREAK. Jeśli zainstalowany zostanie sterownik, który podaje nazwę zgodnie z DOS–em, to polecenie DIR będzie działać bez zarzutu. Jako parametr można podać podobnie jak w DOS–ie maskę np. *.COM, z tym że handler T: nie zwraca na to żadnej uwagi.

CAR

Uruchamia cartridge pod warunkiem, że jest zainstalowany. Jeżeli go nie ma, to program sprawdza, czy zainstalowany jest BASIC, jeżeli jest – następuje uruchomienie interpretera bez skasowania programu BASIC–a. Jeśli interpreter jest odłączony to wyświetla się napis: No cartridge. Dla przykładu: w BASIC–u piszemy DOS, a następnie w TOS–ie CAR, znowu ukaże się znajomy napis – READY. Polecenie CAR ON powoduje włączenie interpretera, bez względu na to, czy włączyliśmy komputer z wciśniętym klawiszem OPTION, czy bez niego. Program BASIC–a zostanie wówczas skasowany. Natomiast instrukcja CAR OFF wyłącza interpreter, dzięki czemu uzyskujemy dodatkowe 8K wolnej pamięci.

COP

Uruchamia podręczny kopier. Składnia polecenia jest następująca:

COP *name*

gdzie name to nazwa pliku przeznaczonego do kopiowania. Na przykład: po **COP QA** następuje odczyt pliku o nazwie QA (jest to prawdopodobnie Quick Assembler). Jeżeli wystąpi błąd to wykonanie polecenia jest przerywane i następuje powrót do interpretera poleceń. Po odczytaniu całego pliku pojawia się pytanie: **File name**. Kopier pyta się o nazwę urządzenia do zapisu oraz o nazwę pliku. Na przykład: C: lub T:NAME. Po dokonaniu zapisu ponawiana jest prośba o podanie nazwy. Wyjść można z tego naciskając klawisz BREAK.

Najważniejszą czynnością każdego systemu jest odczyt i uruchomienie programów. Jako polecenie wypisujemy nazwę programu przeznaczonego do odczytania i uruchomienia. Na przykład T:DEMO lub DEMO albo znak gwiazdki, która zastępuje każdą nazwę. Dla handlerów, które nie zapisują nazw, wystarczy samo naciśnięcie RETURN. Jeśli podczas odczytu stwier-

dzony zostanie błąd, to przerywany jest odczyt i wyświetlany jest komunikat: Error – *nr*, gdzie *nr* jest numerem błędu. Naciśnięcie klawisza BREAK jest traktowane jako rezygnację i następuje bezpośredni powrót bez wyświetlania komunikatu.

W systemie małego ATARI jest przyjęte, że programy odczytywane przez DOS mają dwa pierwsze bajty równe 255 ($FF). Jeśli TOS podczas odczytu nie znajdzie tych dwóch bajtów, wówczas przerwie odczyt i wyświetli komunikat: Not binary file. Ma to na celu zabezpieczenie systemu przed zawieszeniem w wyniku odczytu pliku z danymi.

Jeżeli podczas wykonywania programu procesor natrafi na rozkaz BRK (prawdopodobnie błędnie napisany program lub zły adres po RUN), to nastąpi wejście do systemu poprzedzone wyświetleniem komunikatu: BRK at *adr*, gdzie *adr* jest adresem rozkazu BRK, w wyniku którego nastąpiło przerwanie.

Jeśli chcemy tylko zmienić nazwę urządzenia, wstarczy napisać nazwę wraz z dwukropkiem. Na przykład C:, T:, M:. Nazwa, która nie figuruje w tablicy HATABS, będzie odrzucona.

Ważną częścią TOS-u jest handler T:. Plik zapisywany na taśmie składa się z nagłówka zawierającego dwubajtowy kod oraz jedenastu bajtów nazwy pliku. Sam plik jest zapisywany w 1024-bajtowych blokach. W celu oznaczenia końca pliku zapisywany jest rekord składający się z samych zer.

Otwarcie pliku w języku BASIC wygląda tak:

OPEN *#nr,op,tr,nazwa*

gdzie *nr*, to numer kanału IOCB, *op* – sposób dostępu. Dozwolone są następujące wartości 4, 5 – odczyt danych; 6, 7 – odczyt nazwy (DIR); 8, 9 – zapis danych. Parametr *tr* – określa reakcję handlera na błędy transmisji. Wartość 128 powoduje, w wyniku wystąpienia błędu, podanie numeru w rejestrze Y. Jeśli *tr* ma wartość 0 i przekaz będzie zawierał usterki, wówczas sterownik zażąda ponownego odczytu błędnego rekordu. Błąd sygnalizowany jest brzęczykiem i zatrzymaniem silnika magnetofonu. Należy wtedy cofnąć taśmę co najmniej o jeden rekord i nacisnąć spację. Jeżeli cofniemy więcej, nie musimy się tym przejmować, bo komputer znajdzie błędny rekord (rekordy są numerowane). Próba odczytu będzie ponawiana aż do skutku. Odczyt możemy przerwać naciskając BREAK. Tekst *nazwa* określa nazwę pliku wraz ze specyfikacją urządzenia. Na przykład T:NAZWA, T1:NAZW, T2:NAME. Przy numerze 2 TOS wydłuża sygnał pilotujący, poprzedzający każdy rekord.

Jeżeli nie podamy numeru po T, zostanie domyślnie przyjęty numer 1.

Dla odczytania nazwy pliku dokonujemy otwarcia jednego z bloków IOCB:

OPEN #1,6,0,"T:"

W przeciwieństwie do odczytu danych nie będzie sygnału dźwiękowego. Nazwę (nazwy) odczytujemy instrukcją

INPUT #1,A$

Oczywiście na końcu nie możemy zapomnieć o zamknięciu bloku IOCB. W podobny sposób odczytuje się nazwę w DOS-ie.

Do TOS-u można dołączyć jeszcze dwie komendy. Nazwa komendy składa się z trzech liter oraz znaku EOL. Pod adresem $B44 znajduje się tablica nazw. Pierwsze wolne pole znajduje się jak łatwo policzyć pod $B64. Rozmiar tego pola wynosi osiem bajtów (dwie nazwy). Natomiast wolne miejsce dla wpisania adresu startu znajduje się pod $BED i ma cztery bajty długości. Po wpisaniu nazwy i adresu pozostaje jeszcze zmienić zawartość dwóch komórek. Pod adresem $B02 jest przechowywana długość tabeli instrukcji, której wartość zwiększamy o cztery, natomiast pod $B03 jest podana długość danych wyświetlanych po instrukcji HLP, którą również zwiększamy o cztery. Wszystkie te czynności najlepiej ilustruje podany niżej program.

Pod adresm $F0 i F1 umieszczamy adres uruchomienia, a pod $F2 i $F3 adres nazwy. Po wykonaniu czynności związanych z komendą, sterowanie powinno zostać przekazane pod adres wskazywany przez wektor DOSRUN ($A).

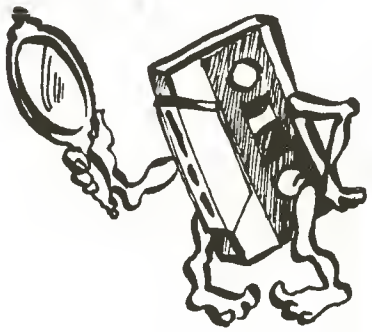
```
      opt %10000


runadr equ $f0
napadr equ $f2
wolne  equ $b63
adres  equ $bee
dlu    equ $b02
po_HLP equ $116b
pocz   equ $6000


      org pocz


* sprawdza czy pola sa wolne

spra ldy #7
sp   lda wolne,y
     bne scon
     dey
     bpl sp
     ldx #0
     beq zapis

* nie wszystkie wolne

scon cpy #4
     bcs nie
     ldx #4
     bne zapis

* oba pola zapelnione

nie  cpx #5
     bcs ret

* wpisanie

zapis txa
     lsr @
     pha
     ldy #0
zap  lda (napadr),y
     sta wolne,x
     inx
     iny
     cpy #4
     bne zap

     pla
     tax
     lda runadr
     sta adres,x
     inx
     lda runadr
     sta adres,x
     clc
ret  rts
```

TOS udostępnia dodatkowo handler urządzenia M:. Jest to typowy RAMFILE, w którym wielkość zapisanego pliku jest taka, jak wielkość pamięci pod ROM-em OS. Plik zapisany do tego urządzenia jest odporny na zimny start (o ile wcześniej nie wyłączyliśmy zasilania). Oczywiście plik przechowywany pod systemem gryzie się z XLF-em i każdym innym programem, ktory wykorzystuje ten obszar pamięci. Najprostsza na to rada, to nie używać sterownika M: w obecności takich programów.

Uwaga: Przy niektórych odmianach sprzętu "turbo" do poprawnego zapisywania może się okazać konieczne wprowadzenie do TOS-u następującej "łaty":

```
1000 DATA 447,141,205
```

Łatę wykonuje się przy pomocy programu Patcher z TA 7/91.



```
HG 1000 REM *------------------,-------*
VN 1002 REM :   autor:  Witold Sciuk   :
FX 1004 REM :  Turbo Operating System  :
YY 1006 REM : (c) 1992 Tajemnice ATARI :
JP 1008 REM *--------------------------*
EI 1010 DATA 0016ee0ae81018a94c850aa911
RM 1020 DATA 850b4cd6fd000720530000000000
FX 1030 DATA 00000000000000000000207f08d4
UD 1040 DATA d5d2c2cfa0cfd0c5d2c1d4c9ce
GE 1050 DATA c7a0d3d9d3d4c5cd889b207f20
FA 1060 DATA 2863292062792057697 46f6c64
PN 1070 DATA 20536369756b9b9b52554e9b4d
EU 1080 DATA 4f549b4d454d9b4341529b434f
TH 1090 DATA 509b4449529b484c509b434c53
ZP 1100 DATA 9b0000000000000000054ba0000
YS 1110 DATA 00000000000000000000000000
TB 1120 DATA 000000004c6f772 03d20202020
LT 1130 DATA 2020204869676820 3d20202020
HK 1140 DATA 209b42524b2061742020202020
CU 1150 DATA 9b4e6f742062696e6172792066
QZ 1160 DATA 696c659b4e6f20636172747269
XR 1170 DATA 6467659b46696c652 06e616d65
IG 1180 DATA 3a209b4572726f72202d202020
HN 1190 DATA 209b151308067713551299 1006
SE 1200 DATA 126f1476125c11d21000000000
YP 1210 DATA c612160fc90f9d0e8f0e0f0ffd
VC 1220 DATA 0ca9ff8dfc0220c90ca90320fc
YD 1230 DATA fd303a20a80ca954204c0c3030
YZ 1240 DATA a0022 0c30d30295 0f09118c8c0
PX 1250 DATA 0bd0f248a93c8d02d38d0306ad
ZR 1260 DATA 01d309018d01d3a9408d0ed4a5
UT 1270 DATA 1009808d0ed26858604c1f0e85
HX 1280 DATA 00a9ff853020ba0d30f2c500d0
NC 1290 DATA f3a980853020c30d30e550e8c9
TW 1300 DATA 3ad0e4608aa2dbd0088aa25dd0
VO 1310 DATA 038aa224cad0fdaa608540a200
NS 1320 DATA a9838d0fd2a9008d1ad0206b0c
DB 1330 DATA a9038d0fd2a90a8d1ad0206b0c
EH 1340 DATA 2c0ed21007cad0dec64010da60
JU 1350 DATA a9348d02d38d03d3a51009808d
DG 1360 DATA 0ed2a9008d0ed48d00d478ad01
QF 1370 DATA d329fe8d01d360ae000b8618ae
UI 1380 DATA 010b8619a90385366048e618d0
YT 1390 DATA 02e619ee040bd003ee050bae04
HL 1400 DATA 0bec060bad050bed070b689009
HT 1410 DATA a2008e040b8e050b60a20160a2
HP 1420 DATA 08851c461ca9838d0fd2a9008d
MF 1430 DATA 1ad0900520700cb00320750ca9
IR 1440 DATA 038d0fd2a90a8d1ad0900520 70
MR 1450 DATA 0cb00320750ccad0d3601820c9
FP 1460 DATA 0ca900aaa871189001e8c8d0f8
UO 1470 DATA e619c63610f0c6196048ee0f0b
FT 1480 DATA ad1fd029020d002853e20a80c20
CR 1490 DATA 340d851a861ba908243e100ba2
DY 1500 DATA 02a5212901d002a2048aa28086
WT 1510 DATA 3e207d0c103920c90ca9582002
GG 1520 DATA 0da93a2002 0dad0f0b2002 0d68
QQ 1530 DATA 2002 0da51a2002 0da51b2002 0d
KD 1540 DATA a000b1182002 0d2c0ed2100dc8
QR 1550 DATA d0f3e619c63610ed4c7e0f6810
JM 1560 DATA 65a928853420c90ca000a9088d
KU 1570 DATA 040ba200ad0fd22910f0108540
AH 1580 DATA 4a8d1ad02c0ed21043e8d0ebf0
BX 1590 DATA 0f8d1ad0a540f0efa900854 0e0
PP 1600 DATA 339006243070c6501fe009b004
OZ 1610 DATA c634d0cce434a919b002a91185
YW 1620 DATA 34661cce040bd0b9a51c2cfe0c
IX 1630 DATA 6050632056f520f8f210092029
UQ 1640 DATA 0ca08060ee0f0b20a80ca95820
EP 1650 DATA 4c0c30ed 20c30d504ecd0f0bd0
XN 1660 DATA ef20c30d5044c9fe8500d0034c
RM 1670 DATA 810f20c30d5036851a20c30d50
IB 1680 DATA 2f851b20c30d30c250269118c8
YD 1690 DATA d0f4e619c63610ee2029 0c2034
FN 1700 DATA 0dc51ad009e41bd005a500a001
GO 1710 DATA 602c100b1095a08f602029 0c2c
XH 1720 DATA 100b108aa0c860a2008118 20d8
WT 1730 DATA 0cd0e1a9fc4c4d0dad050b0d04
VY 1740 DATA 0bd0632c080b10428d070b2c08
RQ 1750 DATA 0b706120c90ca007a920911888
CT 1760 DATA 10f920000c3020a000a99bd118
YE 1770 DATA f00ec8c010d0f7a92091 18c8c0
GZ 1780 DATA 10d0f9a99b9118c88c060bd022
JC 1790 DATA a9c08d080ba000f0e420250e08
KM 1800 DATA c9fef01e28301ac9fcf00a8d07
MK 1810 DATA 0ba0ffb1188d060b20c90ca200
IY 1820 DATA a1182 0d80ca0016068a0886 0a5
LC 1830 DATA 2aa0048c070ba0008c060b8c08
AF 1840 DATA 0b8c89028c040b8c050b8c0f0b
DC 1850 DATA a42b8c100b290e852ac904f010
DM 1860 DATA c908f047c906d007a9808d080b
QE 1870 DATA a0016020000c30fa200810a002
FP 1880 DATA a201bd0004c99bf01cc920f018
BP 1890 DATA c92af014d118d019c8e8c00df0
XG 1900 DATA 0ac99bf006bd00044c660f20c9
BD 1910 DATA 0c2029 0ca00160a0d0602 0c90c
CA 1920 DATA a9ff8dfc02a9042 0fcfd30f020
DB 1930 DATA 0810a9808d8902853e20a80ca9
IA 1940 DATA 0d207d0c101aa93a8d0004a954
RZ 1950 DATA 2002 0da000b900042002 0dc8c0
TY 1960 DATA 0dd0f54c7e0f4c1f0e2c89021 0
AM 1970 DATA f5ad050b0d040bf0152 0c90c18
QO 1980 DATA 65198519a0ffad040b9118ad05
RI 1990 DATA 0b204d0d20c90ca00098a2038e
JT 2000 DATA 050b9118c8d0fbe619ce050b10
BM 2010 DATA f4a9fe4c4d0da001a200a93ad1
EE 2020 DATA 24f001c8b1249d0004e8c8c00d
LH 2030 DATA d0f560a212a9038d0fd2a9ff8d
BR 2040 DATA fc02a9009d6d0bcad0fa866b86
PE 2050 DATA 36204af2c0003021a636c99bf0
TQ 2060 DATA 16297fc9619006c97bb002e91f
RG 2070 DATA 9d6e0be8e011d0dda99b9d6e0b
GC 2080 DATA a001608636a557c91090032094
WX 2090 DATA efa636bd000b300720b0f2e636
GN 2100 DATA d0f2c99bf002297f4cb0f24829
XT 2110 DATA 0f208f1085006 84a4a4a4ac90a
FA 2120 DATA b0036930606936 60a003b9e502
WT 2130 DATA 208210beda0b9d810ba5009d82
SJ 2140 DATA 0b8810ec20ba10a2812060104c
GY 2150 DATA 7411a99b20b0f2a99b4cb0f2a2
GI 2160 DATA 00bd6e0b9d6c0be8e012d0f560
CJ 2170 DATA 202 0f44c7411a900a20186ca9d
ZA 2180 DATA 090b9de002ca10f760a09fcce7
VT 2190 DATA 02a915ede8029008a9158ce702
LS 2200 DATA 8de802a2258e0602a2118e0702
DH 2210 DATA 20de14a9038dda02ad1fd0f005
HK 2220 DATA 2902f00460 4c77e4a24c860aa2
UM 2230 DATA 1186 0bd027686838d868e90208
CQ 2240 DATA 208210 8da20ba6008ea30b2868
EA 2250 DATA e9002082108da00ba6008ea10b
FQ 2260 DATA a2992060102 0de14a2038609a2
WL 2270 DATA ffd8589a86083 0182 0ba10a200
NJ 2280 DATA 8600a600ec030bf00abd110b20
AF 2290 DATA b0f2e600d0ef20ba1020eb13a9
MU 2300 DATA ba8d6d0ba26c206010a5528d0b
MQ 2310 DATA 0b1869028552201f1008ad0b0b
LF 2320 DATA 8552283 0daa23a8e6d0bec6f0b
YC 2330 DATA d01da221ad6e0bdd1a03f007ca
PA 2340 DATA caca10f630bf20c410ad6e0bc9
QU 2350 DATA 9bf0b5d0d9a000a200bd6e0bd9
UT 2360 DATA 440bf008c8cc020bd0f2f022e8
NK 2370 DATA e003f012bd6e0bc8cc020bf014
```

```
WW 238Ø DATA d944Øbf0eda200f0e1ad710bc9
CQ 239Ø DATA 20f006c99bf002a028984a291e
NA 24ØØ DATA 1869de8d04126cde0bad13d0d0
NQ 241Ø DATA 26ac720bc04fd01aac730bc046
AH 242Ø DATA f008c04ed00fa900f002a9018d
OF 243Ø DATA f8032Ød8104c74e4adf803d003
YR 244Ø DATA 6cfabfa2b52060104c7411a201
JM 245Ø DATA bde0029d090bca10f7ad090b0d
SH 246Ø DATA 0a0bf0ea20d6fd6c090bad02d3
EY 247Ø DATA 49088d02d34c74116ce202a250
YW 248Ø DATA a9008d98038d99032056e41003
DD 249Ø DATA 4c38136020eb132Øf513a250a9
YN 25ØØ DATA 039d4203a96c9d4403a90b9d45
KZ 251Ø DATA 03a9069d4a032056e430dba907
QX 252Ø DATA 8d92032Øba10a2008e0b0b2063
DW 253Ø DATA 12ae0b0b9d6e0bc99bf008e8e0
XC 254Ø DATA 128e0b0bd0eba99b9d6e0ba26e
UQ 255Ø DATA 2060104c9f122029142056e430
LC 256Ø DATA a4a9ff851da9008de2028de302
BT 257Ø DATA 8d98038d9903a9408d0b0ba907
HW 258Ø DATA 8d92032063128d94032063128d
RS 259Ø DATA 95032d9403c9ffd00d241d1003
QM 26ØØ DATA 20d810a900851df0c8241d1008
LI 261Ø DATA a2a52060104c74112063128 51e
GF 262Ø DATA 206312851fa207204e143011ad
XQ 263Ø DATA e2020de302f0a32060124cd212
NT 264Ø DATA 4c3c122029Øcc080f035c088d0
RJ 265Ø DATA 0a2c0b0b70ed102a4ca514a900
VA 266Ø DATA aaf81869019001e888d0f7d820
MH 267Ø DATA 82108dd70ba5008dd80b8a2082
QR 268Ø DATA 10a5008dd60ba2ce2060104c4c
BP 269Ø DATA 11ad720bf004c99bd0034c4712
TW 27ØØ DATA a003a9009904Øb8810fac8b972
NL 271Ø DATA 0bc99bf02b48a202bd040b9d05
OX 272Ø DATA 0bca10f768c9309Øcec93a900c
RQ 273Ø DATA c9419Øc6c947b0c2e9361002e9
JL 274Ø DATA 2f8d040bc8c00490cead050b0a
QA 275Ø DATA 0a0a0a0d040b8de913ad070b0a
AQ 276Ø DATA 0a0a0a0d060b8dea13a24cad0f
UC 277Ø DATA d22908d002a26c8ee8136c040b
JZ 278Ø DATA a250a90c9d42034c56e4a200bd
TY 279Ø DATA 720b9d6e0be8e00dd0f56020eb
MT 28ØØ DATA 13a2c22060102Ø1f1030 3dac6e
KP 281Ø DATA 0ba908d0172Øeb13ad710bc99b
YW 282Ø DATA d0058d6e0bf00320f513ac6c0b
KA 283Ø DATA a9048d9a03a903a25Ø8d9203a9
EA 284Ø DATA 00c043d002a9808d9b03a96c8d
QV 285Ø DATA 9403a90b8d95036038a51eed94
LK 286Ø DATA 03a8a51fed9503c8088c9803a8
NQ 287Ø DATA 28d001c88c99038e9203a2504c
ER 288Ø DATA 56e4a9808d0b0b2017142056e4
EY 289Ø DATA 1003 4c38132Ød810a207ace702
FK 29ØØ DATA 8c9403ace8028c9503ace50284
OE 291Ø DATA 1eace602841f204e1430dd2056
TL 292Ø DATA f54c7411ad9803851ead990385
NC 293Ø DATA 1f2003143Øeea51e8d9803a51f
IE 294Ø DATA 8d9903a96e8d94032056e430b4
VS 295Ø DATA a20bade7028d9403ade8028d95
OY 296Ø DATA 032067143Øc74caf14a254a90b
IS 297Ø DATA a0f420bceea24da915a0934cbc
XM 298Ø DATA eea52a290cc904f009c908f001
AV 299Ø DATA 60a9803002a9008d0e0ba9028d
UR 3ØØØ DATA 0c0ba9c08d0d0ba00160ae0c0b
YX 3Ø1Ø DATA 8604ae0d0b86056Øee0c0bd012
GK 3Ø2Ø DATA ee0d0b100cae0d0be0d0d006a2
LZ 3Ø3Ø DATA d88e0d0b602c1e1560 2c0e0b10
RV 3Ø4Ø DATA 1620b70cae0c0bac0d0be8d001
JS 3Ø5Ø DATA c88e00c08c01c020290ca00160
MZ 3Ø6Ø DATA 4820b70c20131568a000910420
UY 3Ø7Ø DATA 290c201e153Øe9a0c0602Øb70c
WD 3Ø8Ø DATA 201315a504cd00c0a505ed01c0
PU 3Ø9Ø DATA b00da000b1042Ø290c201e15a0
SC 31ØØ DATA 01602Ø290ca08860ef1438156d
MV 311Ø DATA 155615Øf0ffd0c0080
```

Witold Ściuk

"Klocki" to gra logiczna będąca skróconą wersją znanego "Tetrisa". Dla tych, którzy nie wiedzą na czym ona polega kilka słów na temat zasad gry.

Podstawową zasadą jest ustawienie jak największej ilości spadających w studni klocków. Klocki te spadają w dół, na co nie mamy wpływu, natomiast można je przesuwać w linii poziomej, a także obracać o dziewięćdziesiąt stopni. Tytuł oryginału wskazuje na liczbę cztery, która określa ilość kwadratowych elementów składających się na poszczególne klocki.

Klocki rozpoczynają spadanie w losowym miejscu najwyższego punktu studni i są obrócone pod różnymi kątami. Szybkość spadania klocka jest zależna od ilości ustawionych już klocków i rośnie w trakcie gry. Aktualna prędkość (poziom gry) wyświetlona jest w dolnej części okna gry. W prawym dolnym rogu wyświetlony jest klocek, który będzie spadał jako następny. Po zapełnieniu dokładnie całej linii zostaje ona skasowana, a wszystkie klocki znajdujące się nad nią zostają przesunięte o jedną pozycję w dół. Za ustawienie klocka przyznawany jest jeden punkt, a za zlikwidowanie linii – dziesięć. Ilość punktów zdobytych przez grającego wyświetlona jest w lewym dolnym rogu pola gry. Sterowanie grą odbywa się za pomocą klawiatury, a znaczenie klawiszy jest następujące:

- **+ i *** – przesuwanie klocka w lewo i prawo.
- **SHIFT** – obrót klocka o dziewięćdziesiąt stopni w prawo.
- **SPACE** – chwilowe zwiększenie prędkości spadania klocka (działa do momentu puszczenia klawisza).
- **CLEAR** – przerwanie gry i powrót na planszę tytułową.
- **ESC** – opuszczenie gry.

```
IJ 1000 REM *--------------------------*
CS 1002 REM : KLOCKI                    :
VB 1004 REM :  autor: Dariusz Zolna     :
YY 1006 REM : (c) 1992 Tajemnice ATARI :
JP 1008 REM *--------------------------*
SG 1010 DATA ffff0088a58e208d8ca9ff8dfc
PL 1020 DATA 0202e8cadfc02c91cd0034c5a
QK 1030 DATA 89a514291fd012a910a21620d0
QA 1040 DATA 8ca004b1b0498091b0881of720
IB 1050 DATA 6589d0d1a004a91091b0881ofb
LS 1060 DATA c6b1a0fe91b0a90ea2012od08c
GB 1070 DATA a5b038e92885b2a5b1e90085b3
SX 1080 DATA a9328d00d2a2128a29010a0a0a
ZX 1090 DATA 09e08d01d2a009a93f91b0e013
KP 1100 DATA b004a90091b28810f18a48a200
SS 1110 DATA a9282oef8ca202a92820ef8ca5
QP 1120 DATA 14186903c514d0fc68aaca10c6
XV 1130 DATA a009a90091b28810fbad0ad229
ZZ 1140 DATA 07c907f0f78d018d20708ba914
MQ 1150 DATA 8d038da9008d048d202e8ca900
RA 1160 DATA 8d00d28d01d28dfd8c20058cad
KX 1170 DATA f102f015adf202cd058df00d29
SL 1180 DATA 3fc9069007c908b003205b8aad
CC 1190 DATA fc02c936d006205d8c4c0088c9
GN 1200 DATA 1cd0034c5a89206589d0080d06
OP 1210 DATA 8df00320ae89adf102f009adfc
MS 1220 DATA 02293fc921f005ce028d1009ad
IN 1230 DATA 038d8d028deefc8c20b08aa90e
PU 1240 DATA a2022od08ca00fb1b0c980d009
UC 1250 DATA 202e8c205d8c4c0088881oeea9
ZG 1260 DATA 8e8dfd8c20058cadf2028d058d
TY 1270 DATA ad078d8d068da9ff8df2024cb8
JQ 1280 DATA 88a9ff8dfc02205d8c6c0a00ad
JJ 1290 DATA 0fd229084a4a4a8d078d60a5b0
ZM 1300 DATA 38e92885b2a5b1e90085b3a009
LN 1310 DATA b1b291b0881of9a5b038e92885
HG 1320 DATA b0a5b1e90085b1a5b238e92885
UZ 1330 DATA b2a5b3e90085b3e8e0149od8a9
VZ 1340 DATA 0a20358c4c2c8ba9c88d00d2a9
PD 1350 DATA ea8d01d2a9008d298d8d2a8da0
TV 1360 DATA 0788b9098dcd298d90038d298d
CW 1370 DATA b90a8dcd2a8d90038d2a8d8888
QL 1380 DATA 10e6a20ea00788b9098d9d198d
CP 1390 DATA b90a8d9d218dca888810efa20e
YX 1400 DATA a00788ad2a8d38fd218d99098d
PC 1410 DATA bd198d990a8dca888810eba900
LI 1420 DATA 8dfe8ca007889848b90a8d186d
EO 1430 DATA fc8c6902aab9098d186dfb8c69
FP 1440 DATA 0e20d08ca000b1b0f005a9ff8d
DF 1450 DATA fe8c68a888881od7adfe8cd001
GJ 1460 DATA 60a20ea00788bd198d99098dbd
LL 1470 DATA 218d990a8dca888810ef60a20f
RL 1480 DATA a001c906d004a20da0ff8eff8c
IF 1490 DATA 8c008da9008dfe8ca007889848
AB 1500 DATA b90a8d186dfc8c6902aab9098d
DG 1510 DATA 186dfb8c6dff8c20d08ca000b1
BO 1520 DATA b0f005a9ff8dfe8c68a888881o
RX 1530 DATA d6adfe8cf0016oadfb8c186d00
WV 1540 DATA 8d8dfb8c60a9008dfe8ca00788
IB 1550 DATA 9848b90a8d186dfc8c6902aab9
BX 1560 DATA 098d186dfb8c690e20d08ca000
QW 1570 DATA b1b0f005a9ff8dfe8c68a88888
SP 1580 DATA 10d7adfe8cd0016ocefc8ca980
JJ 1590 DATA 8dfd8c20058ca9012o358ceee04
KW 1600 DATA 8dad048dc932902ba9008d048d
LL 1610 DATA ad038d38e902c902b002a9028d
WC 1620 DATA 038da90ea21620d08ca91438ed
IC 1630 DATA 038d4a0910a00091b08d8f8ea9
OB 1640 DATA 0ea2012od08ca213a9ff8dfe8c
YB 1650 DATA a009b1b0c980f005a9008dfe8c
ZQ 1660 DATA 8810f2adfe8cf0034c71898a48
RF 1670 DATA a200a92820ef8c68aaca10d5a9
RA 1680 DATA 648d00d2a00f8c01d2202e8c88
YA 1690 DATA 10f7ad018da8b92b8d8d088dad
DJ 1700 DATA 018d0a0a0a18693285b4a90018
HP 1710 DATA 698d85b5a007b1b499098d8810
OZ 1720 DATA f8a9008dfc8ca90938ed088d8d
JJ 1730 DATA fe8cad0ad2290fc90ab0f7cdfe
OG 1740 DATA 8cb0e98dfb8cad0ad22907c907
FP 1750 DATA f0f78d018d0a0a696a85b2a900
PG 1760 DATA 698d85b3a916a2152od08ca201
UM 1770 DATA a001b1b291b0881of98a48a200
JS 1780 DATA a92820ef8ca202a9022oef8c68
XX 1790 DATA aaca10e2ad0ad22903a8984820
HJ 1800 DATA b88968a8881of660a007889848
AK 1810 DATA b90a8d186dfc8c6902aab9098d
HQ 1820 DATA 186dfb8c690e20d08ca000adfd
LO 1830 DATA 8c91b068a888810db60a514c5
FI 1840 DATA 14f0fc6048a910a21620d08c68
JA 1850 DATA aaa004b1b018690191b099918e
HA 1860 DATA c91a900aa91091b099918e8810
TW 1870 DATA e8cad0e360a9a685b2a98e85b3
UU 1880 DATA a90da2002od08ca217a00bb1b2
DX 1890 DATA 91b0881of98a48a200a92820ef
MA 1900 DATA 8ca202a90c20ef8c68aaca10e2
JS 1910 DATA 60a98685b4a98d85b5a9a685b2
XM 1920 DATA a98e85b3a90da2002od08ca217
GM 1930 DATA a00bb1b091b2b1b491b0881of5
SO 1940 DATA 8a48a200a9282oef8ca202a90c
JQ 1950 DATA 20ef8ca204a90c20ef8c68aaca
UG 1960 DATA 10d760186558850a900655985
RP 1970 DATA b1ca100160a5b018692885b0a5
YY 1980 DATA b1690085b14cdb8c1875b095b0
SP 1990 DATA a90075b195b160000000000000
BP 2000 DATA 0000140000000000000000000000
YR 2010 DATA 0000000000000000000000000000
YU 2020 DATA 0000000000000000000000000000
MY 2030 DATA 0000000202020302020101o0000
KE 2040 DATA 0101010201000001000101o201
EX 2050 DATA 0100020000010101000000100
KA 2060 DATA 0200030000000100020102o100
HZ 2070 DATA 0101000201020000010000o101
FD 2080 DATA 01000049cb00004bcb0000049c9
QN 2090 DATA 00590059004f00cb004900cc49
CM 2100 DATA 4f4b4cfc0000000000000000000
SE 2110 DATA 00fcfcd9d9d900c959c9d9d94b
RP 2120 DATA fcfcd959d900595959d959d9fc
RK 2130 DATA fcd9d9d94fcb59cbd9d9d9fcfc
BM 2140 DATA 0000000000000000000000fcfc00
HG 2150 DATA 00637a796c691a0000fcfc0000
BE 2160 DATA 0000000c00000000fcfc2e616a
QR 2170 DATA 6b726f74737a79fcfc00003465
OJ 2180 DATA 747269730000fcfc000000000
NF 2190 DATA 000000000fcfc525252525252
XO 2200 DATA 52525252fcfc0000000000000
SL 2210 DATA 000000fcfc00002175746f721a
GO 2220 DATA 0000fcfc002461726975737a00
PE 2230 DATA 00fcfc000000003a6f6c6e6100
JI 2240 DATA fcfc0000000000000000000fc
IQ 2250 DATA fc236f707972696768741afcfc
AZ 2260 DATA 34212a252d2e2923250ofcfc21
YX 2270 DATA 342132290011191912fcfc0000
FN 2280 DATA 000000000000000fcc9d5d5d5
QV 2290 DATA d5d5d5d5fcd5d5cf5930756e6b
JX 2300 DATA 74791afc0000d959101a101010
NQ 2310 DATA 1010fc0000d9cb555555555555
KM 2320 DATA 55fc5555cce002e1020088
```

Dariusz Żołna

PRAWDZIWI PROGRAMIŚCI

Tekst, którego tłumaczenie znajduje się poniżej, przywędrował do mnie z USA na dyskietce z programami Public Domain. Autorem tego monologu jest anonimowy amerykański programista, prawdopodobnie związany z firmą Motorola. Zauroczony stylem tej wypowiedzi, pokazałem tekst Naczelnemu i tak oto powstała jego polska wersja.

Jarosław Syrylak

Motto: "Prawdziwi programiści nie używają Pascala"

W dawnych, dobrych czasach, zwanych "Złotą Erą" komputerów, łatwo było odróżnić mężczyzn od chłopców (wtedy mówiło się "oddzielić prawdziwych mężczyzn od maminsynków"). W tym okresie prawdziwymi mężczyznami byli ci, którzy rozumieli programowanie komputerów, zaś maminsynkami - ci, którzy nie rozumieli. Prawdziwy programista mówił rzeczy typu: "DO 10 I=1,10" i "ABEND" (oni naprawdę mówili wielkimi literami!), a reszta świata mówiła "komputery są dla mnie zbyt skomplikowane" i "nie potrafię spoufalić się z komputerami - są zbyt bezosobowe". Prawdziwi mężczyźni nie "spoufalają się" z niczym i nie boją się bezosobowości!

Lecz, jak to zwykle bywa, czasy się zmieniają. Patrzymy dziś na świat, w którym starsze panie używają piecyków mikrofalowych wyposażonych w komputery, dwunastoletnie dzieciaki leją prawdziwych mężczyzn w grach typu "Asteroids" i "Pac-Man" i każdy może kupić, a nawet zrozumieć, swój własny komputer osobisty. Prawdziwym programistom zagraża wyparcie przez maturzystów z mikrokomputerami.

Jak widać, istnieje wyraźna potrzeba pokazania różnicy pomiędzy typowym graczem komputerowym z ogólniaka, a prawdziwym programistą. Kiedy różnica ta stanie się jasna, dzieciaki dostaną materiał do przemyślenia i wzór do naśladowania. Łatwiej także będzie wytłumaczyć pracodawcom prawdziwych programistów, dlaczego pomyłką jest zastępowanie tychże pac-manowymi championami, którym nie trzeba tyle płacić.

1. Języki

Najprostszą metodą oddzielenia prawdziwego programisty od tłumu jest zauważenie, jakiego języka programowania używa. Prawdziwi programiści stosują FORTRAN, maminsynki używają Pascala. Nicklaus Wirth, twórca Pascala, udzielił kiedyś wywiadu, podczas którego zapytano go jak wymawia swoje nazwisko? Odpowiedział, że są tu dwie możliwości - albo wymawia się je jak nazwisko, czyli *wirt* (w oryg. Veert - przyp. tłum.), albo jak wartość, czyli *wert* (w oryg. Worth - przyp. tłum.). Każdy może natychmiast sam domyślić się po tej replice, że Nicklaus Wirth jest maminsynkiem. Jedynym mechanizmem, jaki tolerują prawdziwi programiści jest taki, jaki zaimplementowano w kompilatorach FORTRAN-G i FORTRAN-H dla komputera IBM/370. Prawdziwi programiści nie potrzebują do wykonywania swojej pracy wszystkich tych wyszukanych mechanizmów - są naprawdę szczęśliwi, mając dziurkarkę kart, kompilator języka FORTRAN IV i puszkę piwa.

- Prawdziwi programiści realizują przetwarzanie list w języku FORTRAN.
- Prawdziwi programiści realizują operacje na tekście w języku FORTRAN.
- Prawdziwi programiści realizują obliczenia (jeżeli w ogóle zajmują się obliczeniami) w języku FORTRAN.
- Prawdziwi programiści tworzą programy sztucznej inteligencji w języku FORTRAN.

Jeżeli którejś z tych rzeczy nie możesz zrobić w języku FORTRAN, zrób ją w asemblerze, jeżeli zaś nie da się jej zrobić w asemblerze - oznacza to, że nie warto sobie nią zawracać głowy.

2. Programowanie strukturalne.

W ciągu ostatnich kilku lat teoretycy informatyki doszli do czegoś, co nazwali "programowaniem strukturalnym". Twierdzą, że programy są o wiele bardziej zrozumiałe, jeżeli programista używa specjalnych konstrukcji językowych. Oczywi-

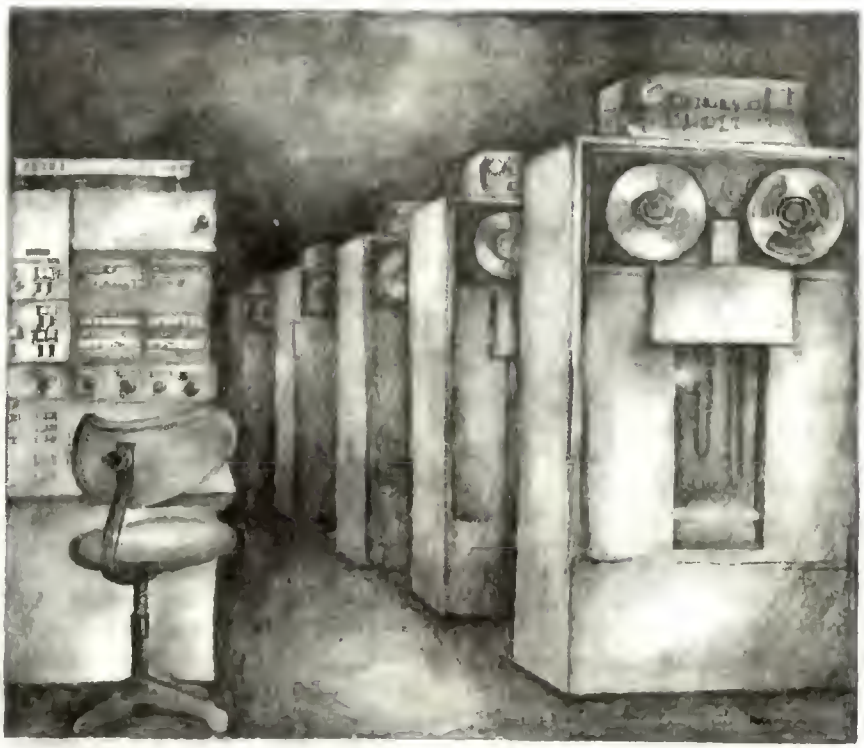
ście, od dawna nie mogą dojść do porozumienia, o jakie konstrukcje im chodzi, więc przykłady, których używają, aby pokazać swój punkt widzenia, niezmiennie trafiają na łamy takich, czy innych czasopism, nikogo nie przekonując!

Kiedy skończyłem szkołę, myślałem, że jestem najlepszym programistą na świecie. Umiałem napisać program, który nie dawał się pokonać w grze w kółko i krzyżyk, używałem pięciu różnych języków programowania i potrafiłem napisać program o długości 1000 wierszy, który działał (naprawdę!). I wtedy wkroczyłem w prawdziwy świat. Moim pierwszym zadaniem było zapoznanie się z programem, napisanym w języku FORTRAN, który miał długość 200000 wierszy, zrozumienie go i przyspieszenie jego działania. Każdy prawdziwy programista powie, że całe programowanie strukturalne na świecie nie może pomóc w rozwiązaniu takiego problemu - potrzebny jest talent.

Kilka obserwacji, związanych z prawdziwymi programistami i programowaniem strukturalnym:

- Prawdziwi programiści nie boją się używać instrukcji GOTO.
- Prawdziwi programiści mogą napisać pętlę DO na pięć stron, nie tracąc wątku.
- Prawdziwi programiści lubią stosować instrukcje IF - one czynią kod ciekawszym.
- Prawdziwi programiści piszą programy, które same się modyfikują, zwłaszcza wtedy, kiedy pozwala to oszczędzić 20 nanosekund w środku krótkiej pętli.
- Prawdziwi programiści nie potrzebują komentarzy - kod jest wystarczająco przejrzysty.
- Dzięki temu, że FORTRAN nie ma instrukcji strukturalnych typu IF, REPEAT ... UNTIL, ani np. CASE, prawdziwi programiści nie muszą martwić się ich nieużywaniem. W zamian uzyskują pożądane rezultaty, stosując odpowiednio rozkazy GOTO.

Wiele pisało się ostatnio także o strukturach danych. Abstrakcyjne typy, struktury, wskaźniki, listy i łańcuchy stały się popularne w pewnych kręgach. Wirth (wspomniany już maminsynek) napisał



nawet książkę, w której usiłuje nas przekonać, że można napisać program oparty na strukturach danych. Jak wiadomo, jedyną pożyteczną strukturą jest tablica. Łańcuchy, listy, rekordy i zbiory są tylko szczególnymi przypadkami tablic i powinny być tak właśnie traktowane, bez zaśmiecania języka programowania wszelkiego rodzaju bzdurami. We wszystkich tych śmiesznych typach danych najgorsze jest to, że musi się je deklarować; w każdym prawdziwym języku programowania typ zmiennej oznacza się pierwszą literą jej sześcioznakowej nazwy.

3. Systemy operacyjne.

Prawdziwi programiści nie używają systemu CP/M, który jest raczej zabawką niż prawdziwym systemem operacyjnym. Nawet starsze panie i dzieci z podstawówki mogą zrozumieć zasady używania CP/M.

Unix jest oczywiście bardziej skomplikowany - typowy użytkownik systemu nigdy nie może spamiętać jaką komendę wyświetlania wywołać w tym tygodniu, ale szczerze mówiąc, Unix to raczej rodzaj gry komputerowej. Nikt nie wykonuje prawdziwej pracy pod kontrolą Unix-a; robi się tylko kawały w sieci UUCP i pisze gry tekstowe oraz prace naukowe.

Nie, prawdziwy programista używa systemu OS/370. Dobry programista znajduje opis błędu IJK3051 w podręczniku użytkownika JCL. Wybitny programista potrafi napisać program bez zaglądania do podręcznika użytkownika systemu. Genialny programista znajduje błąd w sześciomegabajtowym, szesnastkowym wydruku pamięci, bez używania kalkulatora.

OS to niezapomniany system operacyjny. Można np. zniszczyć efekty wielu dni pracy przez umieszczenie jednej spacji w złym miejscu. Najlepiej kontaktować się z systemem przez dziurkarkę kart. Niektórzy ludzie zarzekają się, że na OS/370 działa system z podziałem czasu, ale po dogłębnej analizie problemu doszedłem do przekonania, że są w błędzie.

4. Narzędzia programistyczne.

W teorii prawdziwy programista może uruchamiać swoje programy po zakodowaniu ich na płycie czołowej komputera. Kiedyś, gdy komputery miały jeszcze płyty czołowe, czasem tak się robiło. Typowy prawdziwy programista wprowadzał do pamięci procedury ładowania, gdy zdarzyło się, że program zniszczył kawałek systemu operacyjnego. Legenda głosi, że Seymore Cray, twórca superkomputera Cray

I i większości komputerów firmy Control Data, osobiście wstukał pierwszy system operacyjny dla komputera CDC7600. Nie trzeba chyba wyjaśniać, że Seymore jest prawdziwym programistą.

W niektórych firmach przetwarzanie tekstu nie wymaga już dziesięciu inżynierów, obsługujących dziurkarkę 029. Dzisiaj prawdziwi programiści wykonują swoją pracę przy pomocy "edytorów tekstu". Wielu ludzi uważa, że najlepsze edytory tekstu powstają w centrum badawczym firmy Xerox w Palo Alto dla komputerów Alto i Dorado. Niestety, żaden prawdziwy programista nie będzie używał komputera z systemem operacyjnym, zwanym SmallTalk i rozmawiał (ang. talk oznacza rozmawiać - przyp. tłum.) z komputerem za pomocą myszy.

Kłopot z takimi edytorami polega na tym, że prawdziwi programiści uważają zasadę "widziały gały, co brały" (w niektórych pismach nazywaną WYSIWYG) za niesłuszną, podobnie, jak w stosunku do kobiet. Nie, prawdziwy programista potrzebuje edytora, pracującego w oparciu o inną zasadę: "sam się o to prosiłeś", edytora skomplikowanego, tajemniczego, wszechmocnego, złośliwego i niebezpiecznego.

Prawdziwi programiści wolą w większości wypadków bezpośrednie wprowadzanie kodu binarnego niż robienie poprawek w tekście źródłowym programu. Robią to tak często, że większość programów ma niewiele wspólnego z tekstem początkowym. W wielu przypadkach tak mało, że można powiedzieć, iż tekst źródłowy nie istnieje. Kiedy trzeba poprawić taki produkt, nikt nie odważy się zlecić tej pracy komuś, kto nie jest prawdziwym programistą - żaden maminsynek nie wiedziałby nawet, od czego zacząć. Tę technikę nazywamy utajnianiem naszej pracy.

Oto kilka narzędzi programistycznych NIE używanych przez prawdziwych programistów:

- Preprocesory języka FORTRAN
- Debuggery, pracujące na tekście źródłowym. Wystarczą wydruki pamięci.
- Kompilatory, które sprawdzają zakres zmiennych. Zabijają one inicjatywę i pomysłowość oraz nie pozwalają na modyfikację systemu operacyjnego przez zastosowanie ujemnych indeksów.
- Archiwa tekstów źródłowych. Lepiej przechowywać je na kartach perforowanych.

5. Prawdziwy programista przy pracy.

Żaden prawdziwy programista nie chciałby zostać znaleziony martwy nad programem obliczeniowym w języku COBOL. Prawdziwy programista lubi zadania najwyższej ważności.

- Prawdziwi programiści pracują w Los Alamos National Laboratory, pisząc symulacje bomby atomowej dla komputera Cray 1.
- Prawdziwi programiści pracują dla Narodowej Agencji Bezpieczeństwa, dekodując rosyjskie transmisje.
- To dzięki wysiłkom tysięcy prawdziwych programistów pracujących dla NASA, nasi chłopcy mogli polecieć na księżyc przed Rosjanami. Nikt w takim wypadku nie zaufa programowi w języku Pascal.
- Prawdziwi programiści pracują dla firmy Boeing, projektując systemy operacyjne dla pocisków Cruise.

Wielu prawdziwych programistów pracuje dla rządu Stanów Zjednoczonych, jednak ostatnio nad horyzontem zaczęła się formować czarna chmura. Wygląda na to, że jakieś wysoko postawione maminsynki z Departamentu Obrony zdecydowały, że wszystkie programy wojskowe powinny być pisane w zunifikowanym języku, zwanym ADA. Początkowo wyglądało na to, że jest to oręż wymierzony w prawdziwych programistów - język strukturalny, z typami danych itd. Na szczęście, po pewnej adaptacji, stał się wystarczająco atrakcyjny, bardzo zwięzły, wyposażony w mechanizmy dostępu do pamięci i możliwość modyfikacji systemu operacyjnego. Poza tym Edgar Dijkstra go nie lubi! (jak zapewne wiecie napisał on książkę pt. "Szkodliwe GOTO", będącą kamieniem milowym w metodologii programowania). A w ogóle, to zdeterminowany prawdziwy programista może napisać programy fortranowe w dowolnym języku.

Prawdziwy programista może pójść na kompromis ze swoimi zasadami i pracować nad czymś bardziej trywialnym niż sposoby niszczenia życia, o ile praca ta przynosi odpowiednie zarobki. Paru prawdziwych programistów pracuje w firmie ATARI, tworząc gry komputerowe (oczywiście, nie grają w nie - prawdziwy programista zawsze umie pokonać maszynę - gdzie tu więc współzawodnictwo?). Kilku innych zajmuje się grafiką komputerową, chociaż nie znajduje ona większego zainteresowania. Dobrą stroną tej pracy jest to, że wykonuje się ją w FORTRAN-ie, można więc uniknąć obrzydliwego COBOL-a...

6. Środowisko naturalne prawdziwego programisty.

Typowy prawdziwy programista żyje przed terminalem komputera. Wokół terminala znaleźć można:

- listingi wszystkich programów, nad którymi kiedykolwiek pracował, rozrzucone przypadkowo na każdej płaskiej powierzchni;
- około pół tuzina filiżanek z niedopitą kawą, czasem wypełnionych petami, lub innymi śmieciami;
- podręcznik użytkownika systemu operacyjnego JCL, otworzony na jakiejś interesującej stronie;
- przyklejony do ściany kalendarz na rok 1969 z psem Snoopy, wydrukowany na drukarce znakowej;

- schemat blokowy w szufladzie - pozostałość po poprzednim użytkowniku biura (prawdziwy programista pisze programy, a nie dokumentację).

Prawdziwy programista jest w stanie pracować 30, 40 a nawet 50 godzin bez przerwy. W rzeczywistości, nawet woli pracować w taki sposób. Jeśli nie gonią go terminy, przejawia tendencje do rozważania wielu aspektów jakiegoś niewielkiego problemu przez pierwsze 9 tygodni pracy, a potem kończy całość projektu w ostatnim tygodniu, podczas 50-godzinnego maratonu. To nie tylko robi wrażenie na jego agencie, który prawie załamał się, nie widząc postępów w pracach, ale także jest dobrym wytłumaczeniem dla nienapisania dokumentacji. Ogólnie:

- żaden prawdziwy programista nie pracuje w normalnych godzinach pracy;
- prawdziwy programista nie nosi krawata;
- prawdziwy programista nie nosi modnych butów;
- prawdziwy programista przychodzi do pracy w czasie przerwy obiadowej;
- prawdziwy programista może, ale nie musi znać imienia swojej żony; na pewno natomiast zna na pamięć całą tablicę ASCII!

7. Przyszłość.

Ostatnia generacja programistów bardzo różni się od swoich poprzedników. Większość młodych programistów nigdy nie widziała dużego komputera, wielu z nich nie potrafi nawet liczyć w systemie szesnastkowym bez pomocy kalkulatora. Absolwenci szkół są chronieni przed realiami programowania przez debuggery symboliczne, edytory tekstu, które liczą nawiasy i "przyjazne dla użytkownika" systemy operacyjne. Najgorsze jest to, że wielu młodych naukowców robi kariery w dziedzinie informatyki, nie znając nawet języka FORTRAN. Czyżby nadchodziła era użytkowników UNIX-a i programistów pascalowych?

Z własnego doświadczenia mogę powiedzieć, że przyszłość maluje się w jasnych barwach. Ani OS/370 ani FORTRAN nie wykazują najmniejszych oznak wymierania (mimo wszystkich wysiłków maminsynków na całym świecie!). Nawet najbardziej podstępne sztuczki - jak wprowadzenie programowania strukturalnego do języka FORTRAN - zawiodły! Jasne, niektórzy producenci oferują kompilatory FORTRAN-a 77, ale każdy z nich ma możliwość powrotu do oryginalnego FORTRAN-a 66, żeby kompilować pętle DO, jak w piśmie stoi!

Nawet Unix nie musi być tak straszny jak mogłoby się wydawać. Ostatnia jego wersja ma zalety systemu operacyjnego, godnego prawdziwego programisty. Dwa różne i podstępnie niekompatybilne interfejsy użytkownika, bardzo skomplikowany i tajemniczy sterownik wyświetlania, pamięć wirtualna... Jeśli się pominie jego strukturalność, to nawet język C może być niezły! Nie ma kontroli typów danych, nazwy zmiennych są siedmio- (dziesięcio-?, ośmio-?) znakowe, no i te wskaźniki... Mniam! To tak, jakby najlepsze fragmenty FORTRAN-a i asemblera w jednym miejscu!

Nie, przyszłość nie będzie taka zła. Przez ostatnie kilka lat prasa pisze o tych komputerowych magikach, opuszczających mury uniwersytetu Stanford, czy też M.I.T. i wchodzących w prawdziwy świat. Ponad wszelką wątpliwość, duch Prawdziwego Programisty żyje w tych młodych mężczyznach i kobietach. Dopóki będą istnieć głupie zadania, kretyńskie błędy i nierealistyczne plany - będą prawdziwi programiści gotowi wskoczyć i rozwiązać problem, zostawiając dokumentację na później.

Niech żyje FORTRAN!

Annon. Wersja skrócona: John Wilson i SonOfMotorola

tajemnice ATARI LK Avalon

ogłaszają

COMIESIĘCZNE LOSOWANIE NAGRÓD PREMIOWYCH

dla zamawiających gry i programy w L.K."Avalon". W każdym numerze TA będzie zamieszczany kupon, który upoważnia do wzięcia udziału w losowaniu.

Co miesiąc 100 szczęśliwców,

którzy przyślą oryginalny kupon wraz z zamówieniem (spis programów na ostatniej stronie), otrzymą jedną z zamawianych pozycji

za darmo.

Pozostałe zamówienia będą realizowane w normalnym trybie.

Termin nadsyłania zamówień z kuponami z numeru 4 TA mija 31czerwca.

kupon premiowy ważny do 31 czerwca tajemnice ATARI

L.K. Avalon

oferujemy dla Twojego Atari:

- ☐ **ROBBO** - wspaniała gra zręcznościowo-logiczna. Cena: 29.000 zł.
- ☐ **MISJA, FRED**
  - Zmagania komandosa w tajnej MISJI;
  - Wędrówka z FREDEM przez pełną zagadek i niebezpieczeństw krainę.
  Cena zestawu: 39.000 zł.
- ☐ **LASERMANIA, ROBBO KONSTRUKTOR**
  - Efektowna gra logiczna polegająca na pokonaniu labiryntu komnat przy pomocy umiejętnego sterowania promieniem lasera;
  - Program użytkowy pozwalający na tworzenie gotowych, samodzielnie działających gier.
  Cena zestawu: 39.000 zł.
- ☐ **AUTOMAT PERKUSYJNY**
  - doskonały do tworzenia "muzyczek" komputerowych. Sprzedawany wraz z instrukcją użytkownika. Cena programu: 42.000 zł.
- ☐ **SAPER** - gra strategiczna.
  Cena gry: 37.000 zł.
- ☐ **A.D.2044** - wspaniała gra przygodowa z tekstem.
  Cena gry na dysku: 37.000 zł.
  Cena gry na kasecie: 39.000 zł.
- ☐ **CHAOS MUSIC COMPOSER**
  - pakiet zawiera program do komponowania, moduł odtwarzający (można go dołączyć do własnych programów) oraz wyczerpującą instrukcję użytkownika.
  Cena pakietu 42.000 zł.
- ☐ **PANTHER**
  - wielofunkcyjny edytor tekstów pozwalający drukować po polsku na wszystkich popularnych drukarkach. Sprzedawany także ze złączem MICROPRINT do drukarki.
  Cena bez złącza: 42.000 zł.
  Cena pakietu ze złączem: 220.000 zł.
- ☐ **HANS KLOSS**
  - gra przygodowo - zręcznościowa. Poszukiwania planów tajnej broni w bunkrze Wolfschanze. Cena gry: 37.000 zł.
- ☐ **TRIX**
  - wciągająca gra zręcznościowo - logiczna. Niezły pomysł i dobre wykonanie. Recenzja TA 3/92. Cena gry: 37.000 zł.

Najlepszy na "małe" Atari zestaw do programowania w języku asemblera.

- ☐ QUICK ASSEMBLER
- ☐ BUG HUNTER
- ☐ XL FRIEND - edytor tekstów, tabela kodów, kalkulator i monitor pamięci
- ☐ Zestaw procedur bibliotecznych
- ☐ Podręcznik do wymienionych programów

Cena zestawu: 62.000 zł.

Wymienione niżej programy nie powstały w naszej firmie. Zachęcamy jednak do ich zakupu.

- ☐ **MIECZE VALDGIRA** - przygodowa gra fantasy z doskonałą muzyką. Cena 39.000 zł.
- ☐ **BATTLE SHIPS** - kto z nas nie grał w "okręty"? Tutaj zasady te same, lecz przeciwnik i oprawa znacznie lepsza. Cena 39.000 zł.
- ☐ **TacTic** - gra logiczna, recenzja TA 2/92. Cena 39.000 zł.
- ☐ **KOLONY** - gra ekonomiczna, recenzja TA 2/92. Cena 39.000 zł.
- ☐ **GEOGRAFIA** - prosty program edukacyjny obejmujący niektóre zagadnienia geografii gospodarczej Polski. Cena 35.000 zł.
- ☐ **ORTOGRAFIA** - któż nie ma z nią kłopotów? Ten program może pomóc i wiele nauczyć. Cena 39.000 zł.
- ☐ **VIDEOGRAPH** - proste narzędzie do tworzenia czołówek filmów VIDEO wraz z instrukcją użytkownika. Cena 43.000 zł.
- ☐ **KULT** - najnowsza gra zręcznościowa. Pomysł jakby znany (ZYBEX), ale wykonanie o wiele lepsze. Cena 42.000 zł.
- ☐ **MÓZGPROCESOR** - wznowienie znanej gry tekstowej. Cena 42.000 zł.
- ☐ **MASTER HEAD** - doskonała gra logiczna z kilkunastominutową muzyką. Cena 42.000 zł.
- ☐ **RODERIC** - dawno zapowiadane przygody rycerza w starym zamczysku. Cena 39.000 zł.
- ☐ **BERTYX** - zmagania z wrogim komputerem na planecie BERTYX. Cena 39.000 zł.
- ☐ **OPERATION BLOOD** - Krwawe starcie z niezliczoną ilością przeciwników. (Jakby znajome - Operation Wolf). **Tylko na dyskietce!** Cena 40.000 zł.
- ☐ **MONSTRUM** - popularny robal w nowej oprawie. Cena 40.000 zł.
- ☐ **SKARBNIK** - gra logiczna. Cena 40.000 zł.
- ☐ **I. Q. MASTER** - gra zręcznościowo-logiczno-edukacyjna. Cena 40.000 zł.
- ☐ **HUMANOID** - wspaniała gra zręcznościowa. Cena 42.000 zł.
- ☐ **PONG** - gra zręcznościowa. Cena 40.000 zł.
- ☐ **TANKS** - walka czołgów w labiryncie. Cena 40.000 zł.

PRZYJEMNEJ ZABAWY!

Uregulowanie należności następuje przy odbiorze przesyłki. Zamówienia **na kartach pocztowych**, które powinny zawierać wyszczególnienie **rodzaju nośnika** (kaseta lub dysk) oraz **dokładny adres** zamawiającego, prosimy kierować do:

L.K. AVALON
skr. poczt. 46
38-100 Strzyżów